MANUAL BÁSICO DE PROGRAMACIÓN EN HTML Y CSS

Contenido

1. FRONTEND

Un desarrollador front-end es aquel que maneja toda la parte visual de un proyecto web, es decir, toda parte que entra en interacción con el usuario mediante el cliente (navegador).

Por tanto, este perfil de desarrollador deberá enfocarse en los siguientes aspectos:

-Interacciones

-Animaciones

-Estilos

-Navegación

1.1 Estándares de desarrollo

En el front encontramos tres estándares que son fundamentales para el desarrollo web, estos son:

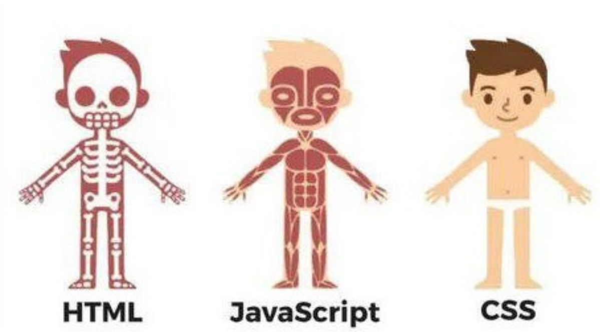
1. HTML: un lenguaje de referencia, o como sus siglas lo indican, un "Lenguaje de Marcas de Hipertexto" (*HyperText Markup Language*), que permite estructurar un documento mediante etiquetas.

2. CSS: un lenguaje utilizado para definir el diseño de un documento previamente escrito en un lenguaje de referencia. Sus siglas indican "Hojas de estilo en cascada"

(*Cascading Style Sheets*), debido a que es un código que se lee por el navegador de arriba hacia abajo.

3. JavaScript: un lenguaje de programación que indicará al navegador que tareas deberá realizar. Es un lenguaje de programación basado en prototipos, por tanto, se podría definir como orientado a objetos.

Este manual solo abarcará los dos primeros estándares, es decir, **HTML** y **CSS**. Sin embargo, es importante comprender la importancia de cada uno de ellos en la estructura de todo proyecto web.



La anterior imagen sirve como ilustración de la funcionalidad de cada uno de los lenguajes anteriormente descritos.

HTML servirá para establecer la estructura de nuestra página web. (Esqueleto)

CSS servirá para definir el estilo o apariencia de nuestra página web. (Apariencia)

JavaScript servirá para añadir dinamismo y una lógica a nuestra página web. (Lógica)

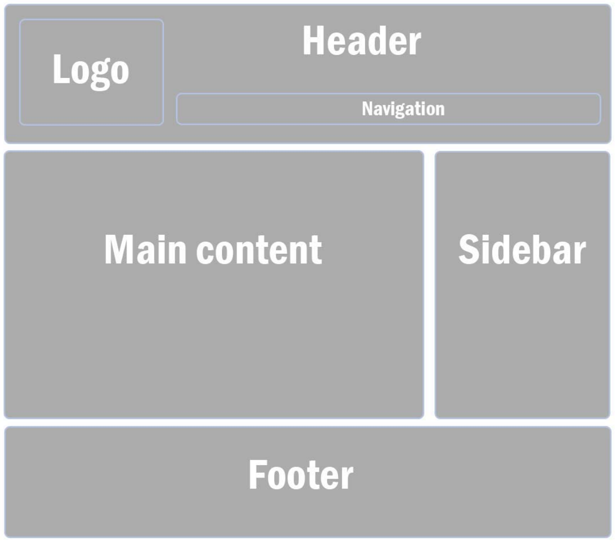
2. HTML

2.1 Anatomía de una página web

La anatomía de una página web son los elementos básicos que la conforman. Estos son:

- Header
    - Logo
    - Nav
- Main content
- Side Bar
- Footer

Estos elementos se pueden ver distribuidos de la siguiente manera:



2.2 Index y su estructura básica: <head>

El head es aquella parte de la estructura del *index.html* que va a llamar a todos los archivos importantes que necesita la página web para su correcto funcionamiento y que no deben ser visibles por los usuarios. Algunos de estos archivos pueden ser:

-Frameworks: son estructuras o plantillas que pueden ser incorporadas en proyectos personales para optimizar los tiempos de trabajo.

-Librerías: Archivos con código previamente escrito, que puede ser usado en otros proyectos.

-Estilos: Archivos CSS con estilos escritos.

-Fuentes: fuentes de letra externas.

-APIs: mecanismos que van a permitir la comunicación entre dos tipos de software diferentes.

2.2.1 Ejemplo de un <head>:

```
<!DOCTYPE html>
<!--Le decimos al navegador que este archivo es del tipo html:5-->

<html lang="es">
  <!--Es la etiqueta "padre" donde vivirá nuestro proyecto. El atributo lang
establece el idioma del sitio web.
    Debemos usarlo para que el navegador pueda traducir nuestra página-->

  <head>

    <meta charset="UTF-8" />
    <!--Este atributo nos ayuda a la hora de incluir caracteres especiales y
emojis en nuestro proyecto-->
```

```
    <meta name="description"content="Descripción aquí" />
    <!--Muestra una descripción de nuestro sitio en los buscadores-->

    <meta name="robots"content="index,follow" />
    <!--Le dice a los robots de los navegadores que rastreen nuestra página
y la muestran en las búsquedas-->

    <title>Mi página</title>
    <!--Título de nuestra página, no confundir con los H1-H6. Este titulo es
el que ves en la pestaña del navegador-->

    <meta name="viewport"content="width=device-width, initial-scale=1.0" />
    <!--Nos ayuda a trabajar en proyectos reponsive-->

    <link rel="stylesheet"href="./css/style.css">
    <!--Linkea/Enlaza archivos de estilos u otros archivos que necesitemos
en nuestro proyecto-->

  </head>
```

*El primer archivo que carga cualquier sitio web será el *index.html,* por tanto, el nombre debe ser siempre el mismo.

2.3 Index y su estructura básica: <body>

El body es aquella parte de la estructura del *index.html* que organizará el contenido que los usuarios ven. En el body se manejan dos tipos de etiquetas, las de contenido y las contenedoras. Las de contenido son las encargadas de mostrar texto, imágenes, videos, etc... Mientras que las contenedoras son las que nos van a permitir darle un orden apropiado a ese contenido. Un ejemplo de etiqueta contenedora podría ser el <div></div>, pues esta etiqueta guarda otras etiquetas de contenido.

2.3.1 Ejemplo de un <body>:

```
<body>

    <header><!--Sección superior de nuestro website-->

      <nav></nav><!--Sección de navegación de nuestro website, siempre
dentro del header-->

    </header>

    <main><!--Main es el contenido central de nuestro website, "la parte del
medio"-->
```

```html
      <section>
        <!--Nuestro website puede estar divido por secciones, esto facilita
mucho el trabajo de organización-->

        <article>
          <!--Contenido independiente de la página. Es reutilizable-->
        </article>

      </section>

      <ul><!--Lista desordenada: Sin numerar-->

        <li><!--Item List. Elementos de la lista--></li>

      </ul>

      <ol></ol><!--Lista ordenada: Numerada-->

    </main>

    <footer><!--Sección final de nuestro website-->

    </footer>

    <p>Soy un texto</p><!--Párrafo, texto-->

    <h1>Soy un titulo</h1>
    <!--Títulos, muestran el texto más grande y con negrilla. Existen desde
el h1 al h6-->

    <a href="#">Soy un link</a>
    <!--Enlaces/links que nos permitirán movernos entre páginas.-->

  </body>
```

2.4 Anatomía de una etiqueta HTML

Las etiquetas HTML tienen una estructura básica. Algunas etiquetas tendrán que cerrarse una vez se llaman, esto se hace con un “/”, de esta manera: <p>Aquí va el contenido que aloja la etiqueta</p>

Otras etiquetas no tendrán su etiqueta de cierre, como los pueden ser las etiquetas de imagen.



2.5 Etiquetas multimedia
Etiquetas utilizadas para insertar imágenes y vídeos.

2.5.1 Etiqueta <img>, <figure> y <figcaption>

```
<figure style="margin: 0;">
    <img src="./pics/small.jpg" alt="Descripcion de la imagen" />
    <figcaption>Imagen</figcaption>
</figure>
```

src: donde está la imagen, es decir, su ruta de acceso. Puede ser desde nuestras carpetas o desde un enlace de la web.

alt: Texto alternativo por si no se renderiza la imagen.

<figure>: genera un contenedor para la imagen.

<figcaption>: agrega una descripción a la imagen. Será visible en la parte inferior de esta misma.

Es recomendable siempre utilizar la etiqueta <figure> para insertar un archivo multimedia.

2.5.2 Etiqueta <video>
Nos permite subir un video mediante la misma forma que una imagen, mediante el atributo src.

```
<section>
    <video controls preload="auto">
        <source src="./video.m4v#t=10,60" />
        <source src="./video.mp4#t=10,60" />
        <source src="./video.gif" />
    </video>
</section>
```

controls: aparecen los controles para manipular el vídeo. No recibe valores, es un parámetro vacío.

preload: ayuda a que el video se empiece a descargar una vez se abre el navegador en esa página.

<source>: va dentro de la etiqueta video y se usa para especificar varias rutas en caso de que el navegador no entienda algún formato de vídeo. Al poner varios formatos, el navegador usa el que más le convenga. Para usarlo se elimina el atributo src del video y se le pone a las etiquetas source. Al video se le dejan los demás atributos.

Si queremos que el vídeo inicie y termine en un minuto/segundo específico, debemos usar unos atributos dentro del src:

#t=: indica el tiempo en el cual empezará y terminará.

10, 90: son los valores en segundos. Donde 10 (izquierda) es donde inicia y 90 (derecha) donde finaliza. Deben ir separados solo por una coma sin espacio.

2.6 Poniendo en práctica lo aprendido

Con todo lo visto en la sección anterior, deberías de tener las suficientes herramientas para elaborar tu primer documento en HTML.

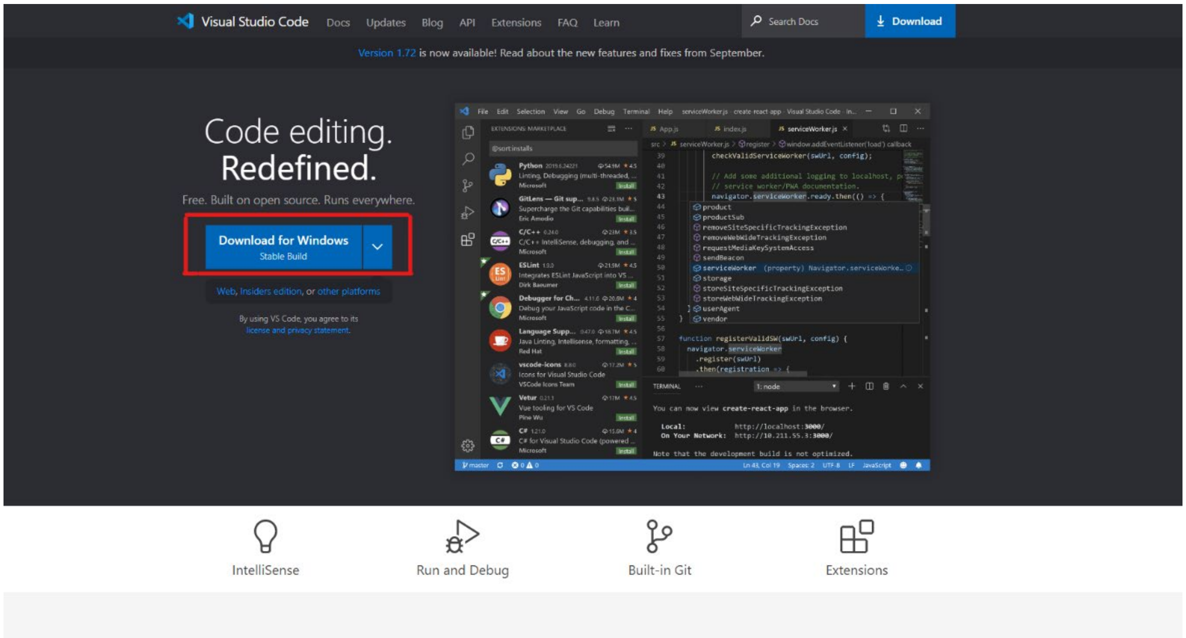
Para practicar, dedicaremos este último apartado para elaborar un documento HTML con todos los componentes que hemos visto hasta el momento.

2.6.1 Configurando nuestro ambiente de trabajo

Para escribir un documento de referencia, podemos hacer uso de cualquier editor de texto que queramos. Sin embargo, existen muchos editores de código fuente que nos facilitarán el trabajo a la hora de escribir. Uno de ellos bastante conocido es *Visual Studio Code* desarrollado por Microsoft.

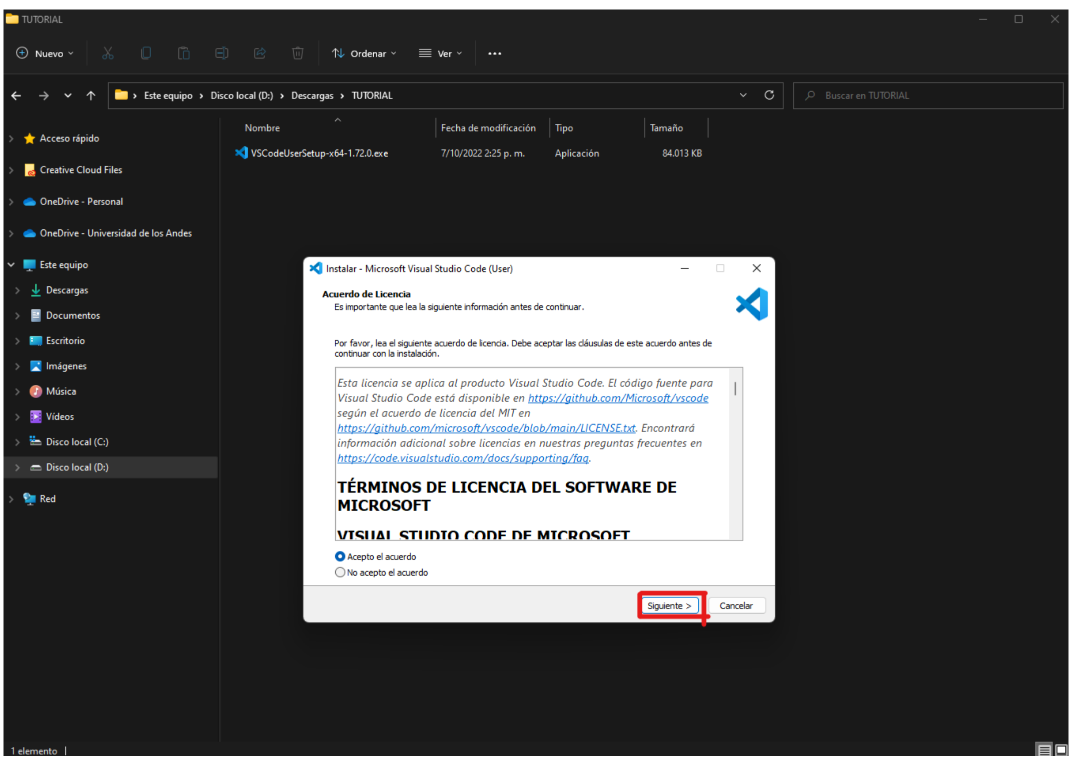
Para instalarlo tendremos que dirigirnos a la web oficial de descarga: https://code.visualstudio.com/

Allí daremos click en el botón "*Download for Windows*" en caso de que trabajemos en dispositivo Windows, o "*Download* for macOS" en caso de que trabajemos en dispositivo Mac.

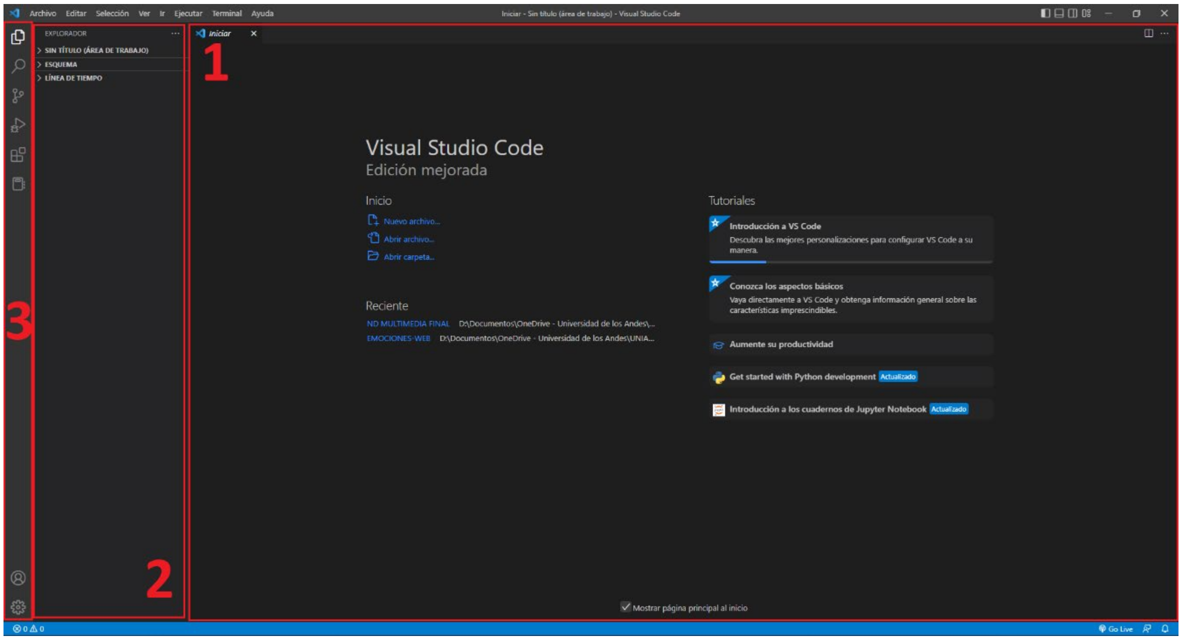


Se descargará un archivo ejecutable .exe en nuestro equipo. Al ejecutarlo saldrá una ventana de instalación.

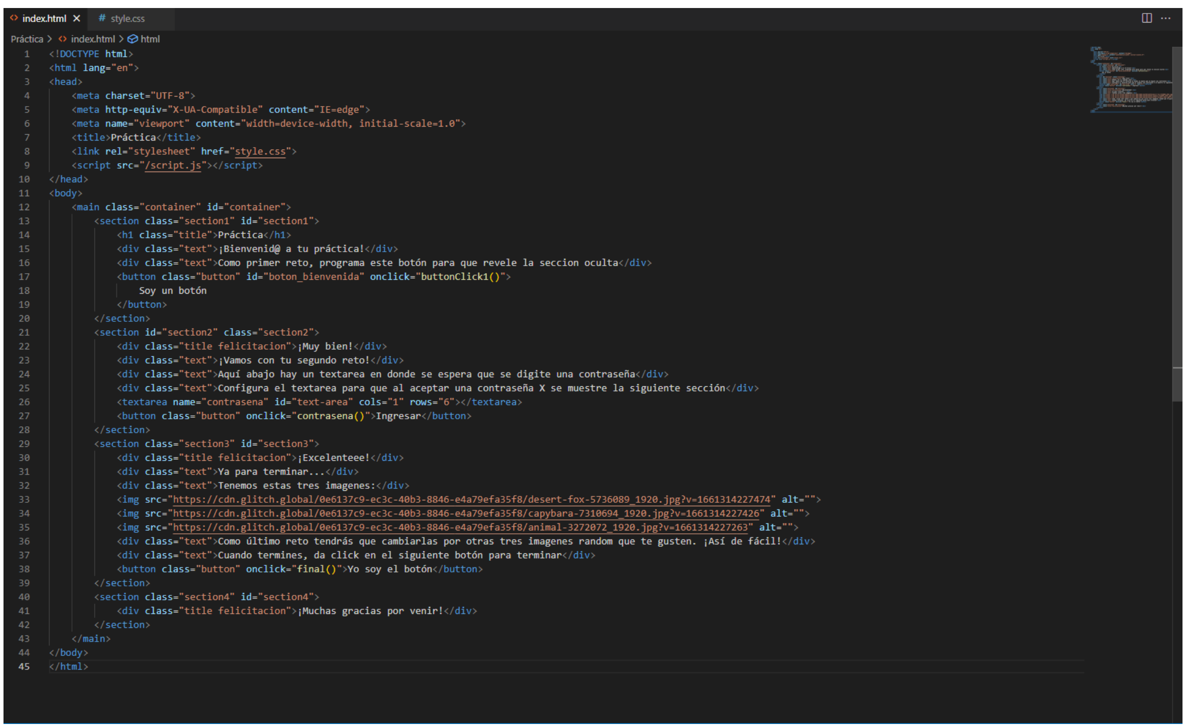
De ahí en adelante es tan sencillo como dar click en “siguiente” e “instalar”.



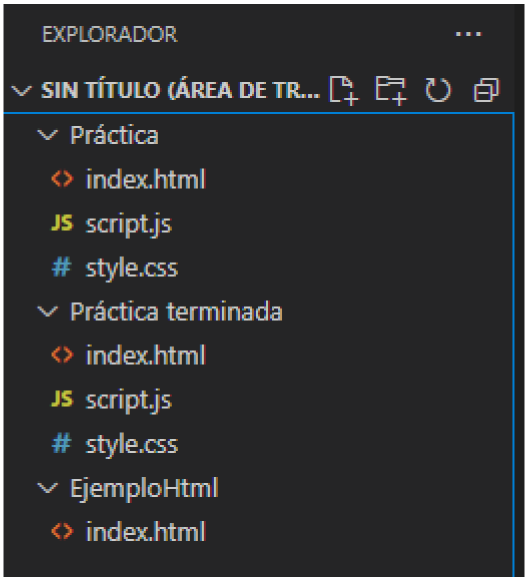
Al finalizar la instalación y abrir el programa, nos encontraremos con la interfaz de *Visual Studio Code*. Hagamos un repaso rápido de cada uno de sus apartados:



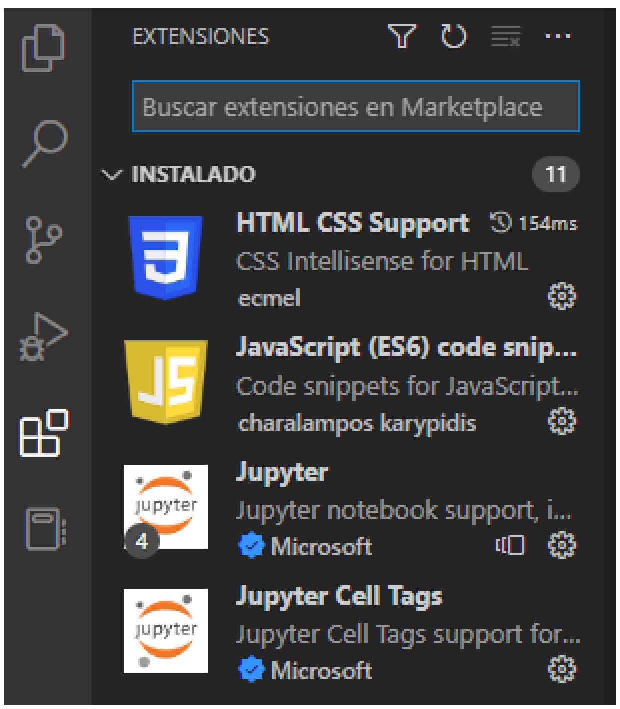
En el apartado **(1)** visualizaremos cualquier documento que tengamos abierto. Es en donde podremos editar y visualizar código.



En el apartado **(2)** tendremos un área de trabajo, parecido a un explorador de archivos. Podremos agregar carpetas de proyectos y crear archivos dentro de ellas.

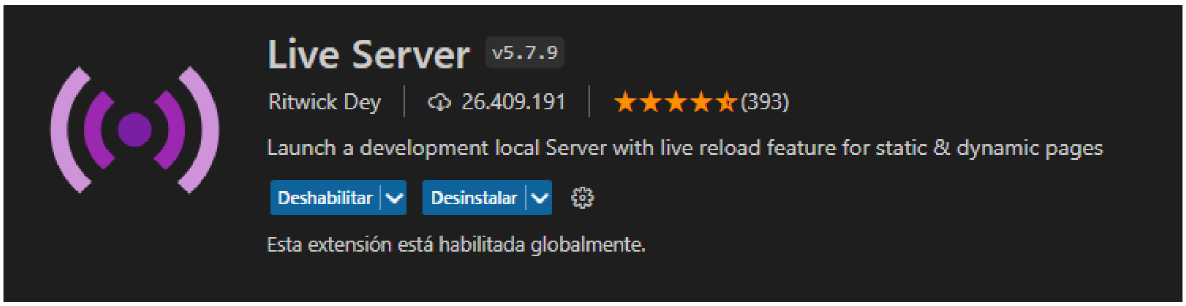


En el apartado **(3)** tendremos acceso a varias herramientas dentro de VS Code. Entre ellas, la más importante es la de "Extensiones". Desde allí podremos instalar distintas extensiones de la comunidad, que nos ayudarán a escribir código mucho más rápido.



Para este apartado de la guía se utilizarán las siguientes extensiones:

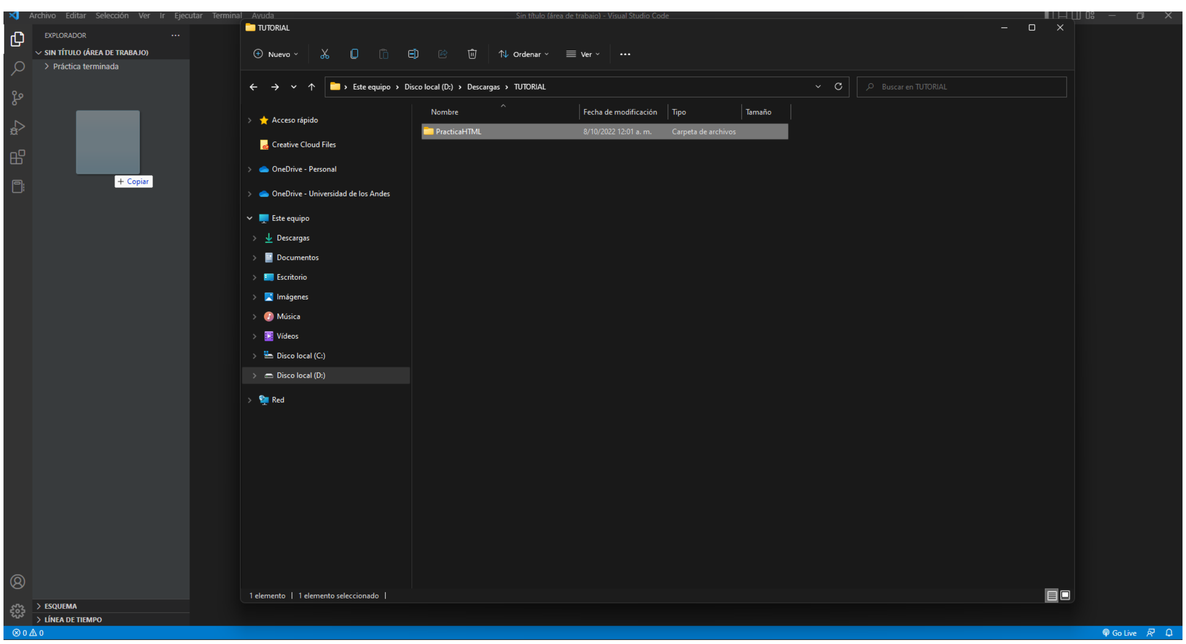




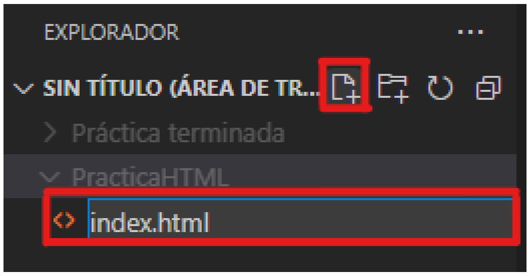
“HTML CSS Support” añadirá funciones de autocompletado de código, mientras que “Live Server” nos permitirá visualizar nuestro proyecto en tiempo real desde el navegador.

2.6.2 Creando nuestro proyecto

Con VS Code instalado, ya podremos empezar a escribir nuestro documento HTML. Lo primero que haremos será crear una carpeta donde alojaremos todos los archivos del proyecto. Lo podemos hacer desde el explorador de archivos del sistema o directamente desde VS Code.



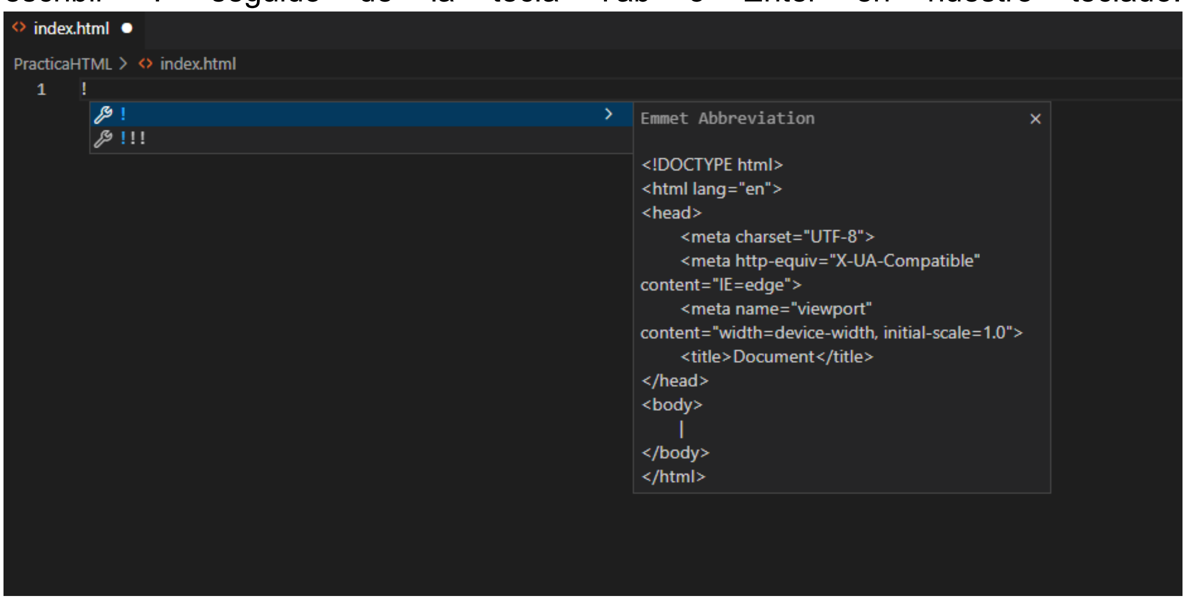
Ahora crearemos nuestro *index.html*. Lo podemos hacer desde el explorador de VS Code.



Al abrir este archivo desde VS Code, ya podremos visualizarlo y empezar a escribir.

2.6.3 Escribiendo nuestro *index.html*

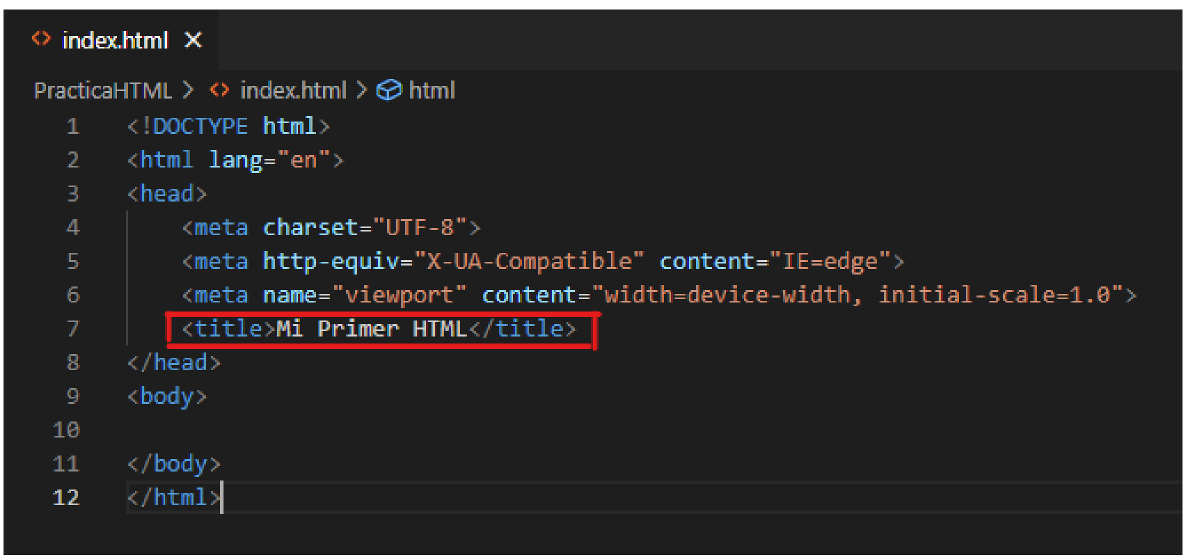
Todo documento HTML tiene una estructura básica (véase apartados 2.2 y 2.3). Escribir esta estructura puede resultar bastante tedioso, por suerte, VS Code viene con algunos snippets que la escribirán por nosotros. Para hacerlo, deberemos escribir "!" seguido de la tecla Tab o Enter en nuestro teclado.



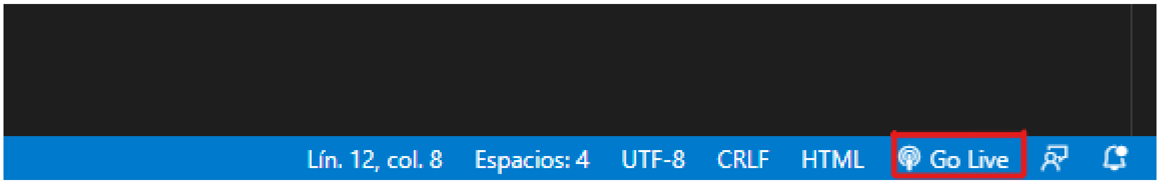


Como podemos observar, ya se habrán escrito varias etiquetas importantes, entre ellas: <head> y <body>.

Recordemos que la etiqueta <title> guarda el nombre que muestra la página en la pestaña del navegador, podemos cambiarla antes de empezar a añadir los demás componentes.

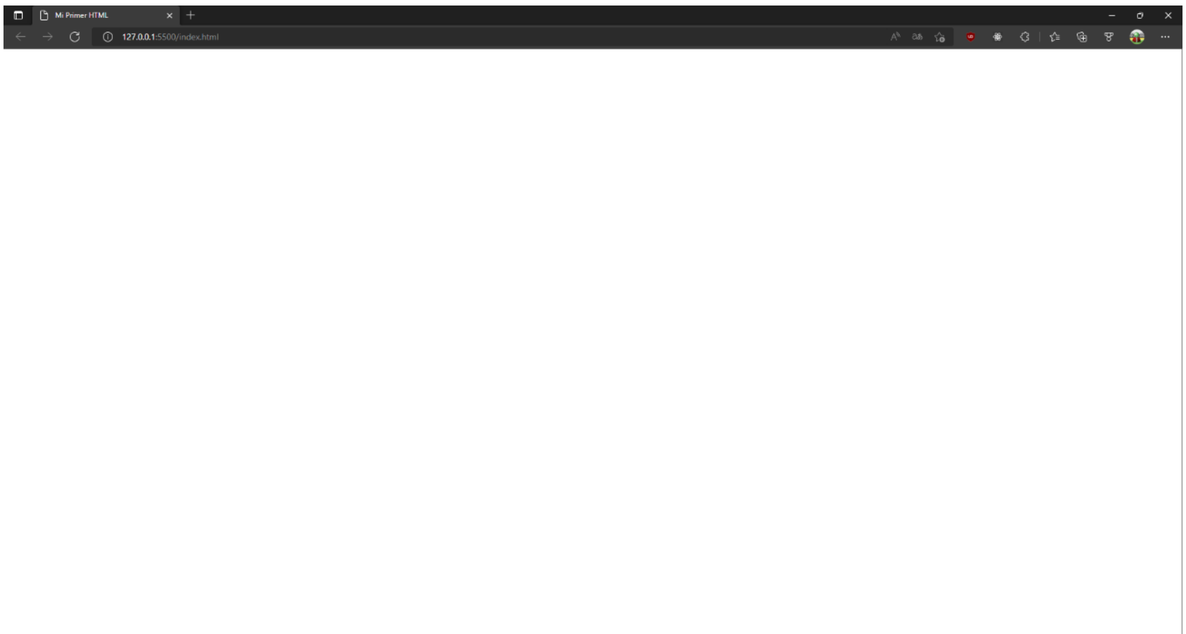


Si bien todavía no hemos añadido nada al <body>, ya podremos visualizar nuestro documento en el navegador. Para ello haremos uso de “Live Server”, dando click en el siguiente botón:



*También podemos utilizar el atajo de teclado Alt + L + O

Al hacerlo, se abrirá el navegador predeterminado del sistema con nuestro documento en blanco.



2.6.4 Antes que nada, la estructura

Antes de empezar a añadir componentes al <body>, es conveniente y una buena práctica empezar a estructurar nuestro documento, dejando en claro la división de los elementos básicos que lo conformarán.

En este caso, dividiremos el contenido del documento en tres secciones principales:

-Sección de bienvenida.

-Sección de imágenes.

-Sección de videos.

Para escribirlo en nuestro *index.html,* podemos hacer uso de las etiquetas <main> y <section> (véase apartado 2.3.1).

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Primer HTML</title>
</head>
<body>
    <main>
        <!-- Sección de bienvenida -->
        <section></section>

        <!-- Sección de imágenes -->
        <section></section>

        <!-- Sección de videos -->
        <section></section>
    </main>
</body>
</html>
```

Como podemos observar, hemos añadido tres etiquetas contenedoras <section> dentro de otra etiqueta contenedora <main>. Nótese que todas ellas están dentro de la etiqueta <body>.

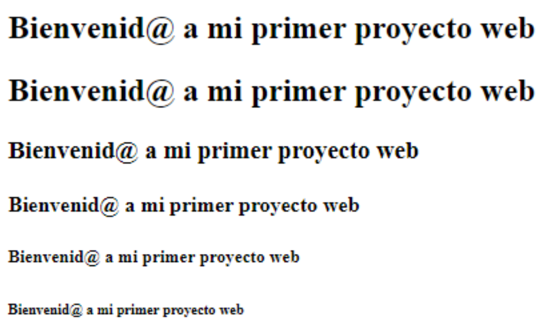
Con la estructura ya planteada, podremos empezar a añadir todos los componentes de forma ordenada.

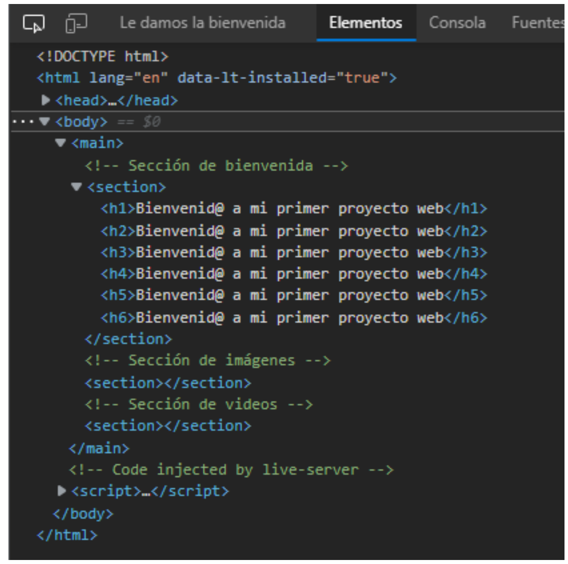
2.6.5 Sección de bienvenida

Vamos a empezar con un título de bienvenida.

Recordemos que podemos utilizar las etiquetas destinadas para ello, estas son: <h1>, <h2>, <h3>, <h4>, <h5> y <h6>.

De tener texto, cada una de ellas lo añadirá en negrilla con un tamaño predeterminado, que por ahora no tocaremos.



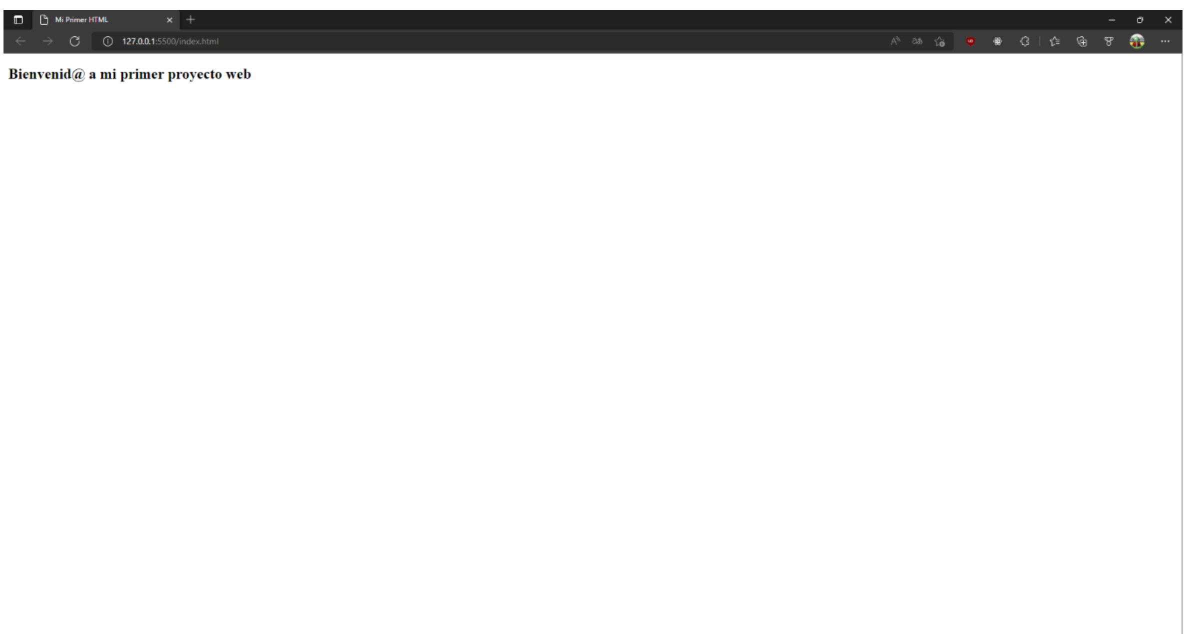


En nuestro caso, optaremos por la etiqueta <h1> para el título.

Dentro de la primera etiqueta <section> escribiremos lo siguiente:

```
<h1>Bienvenid@ a mi primer proyecto web</h1>
```

Al guardar el archivo con Live Server ejecutándose (esto lo podemos hacer con el atajo: Ctrl + S), deberíamos de poder ver los cambios desde el navegador.

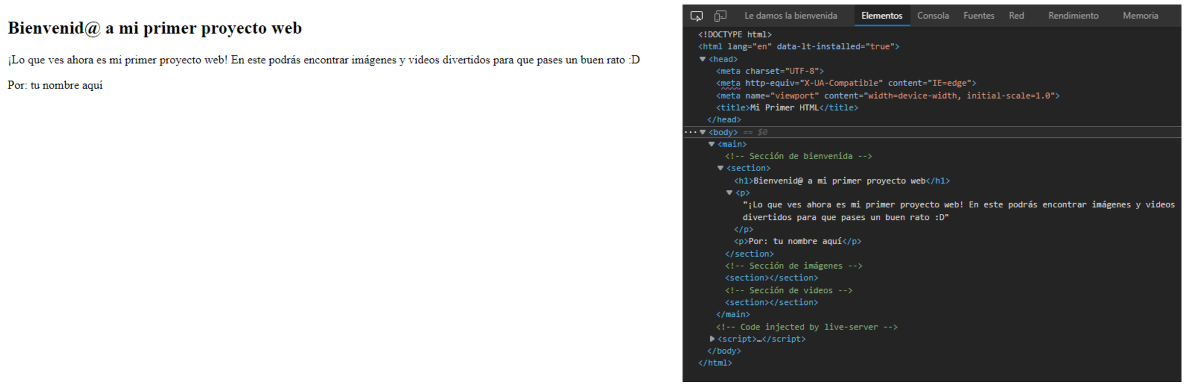


Sigamos añadiendo componentes.

Ahora añadiremos un párrafo de bienvenida seguido de nuestro nombre. Para esto podemos hacer uso de la etiqueta <p>. Escribiremos las siguientes líneas después de la etiqueta de título:

```
<p>¡Lo que ves ahora es mi primer proyecto web! En este podrás encontrar imágenes y videos divertidos para que pases un buen rato :D</p>
<p>Por: tu nombre aquí</p>
```

Al guardar, nuestro documento debería verse de la siguiente manera:



Tip adicional: de querer añadir un párrafo muy largo, añadirlo en una sola línea puede resultar tedioso. Para tener un código más organizado y fácil de leer, podemos utilizar el format de VS Code (Alt + Shift + F).



DESPUÉS

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Primer HTML</title>
</head>

<body>
    <main>
        <!-- Sección de bienvenida -->
        <section>
            <h1>Bienvenid@ a mi primer proyecto web</h1>
            <p>Soy un párrafo muy largo. Soy un párrafo muy largo. Soy un párrafo muy largo. Soy un párrafo muy largo.
                Soy un párrafo muy largo. Soy un párrafo muy largo. Soy un párrafo muy largo. Soy un párrafo muy largo.
                Soy un párrafo muy largo. Soy un párrafo muy largo.</p>
        </section>

        <!-- Sección de imágenes -->
        <section></section>

        <!-- Sección de videos -->
        <section></section>
    </main>
</body>

</html>
```

2.6.6 Sección de imágenes

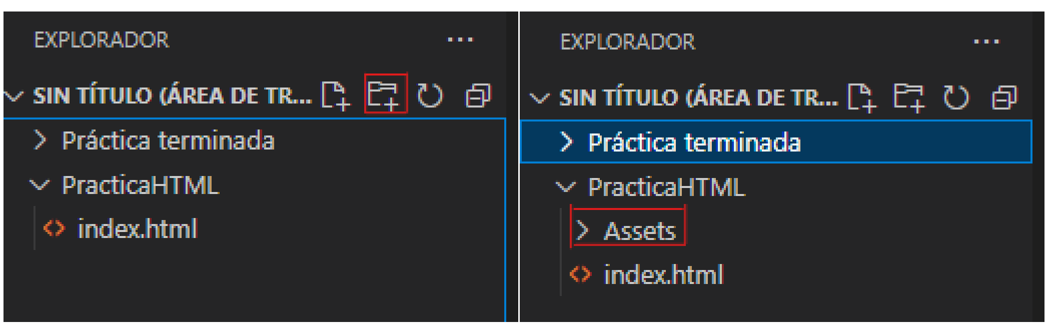
Empezaremos añadiendo un título a la sección.

Esta vez lo haremos con la etiqueta <h3>, para tener un tamaño de fuente más pequeño:

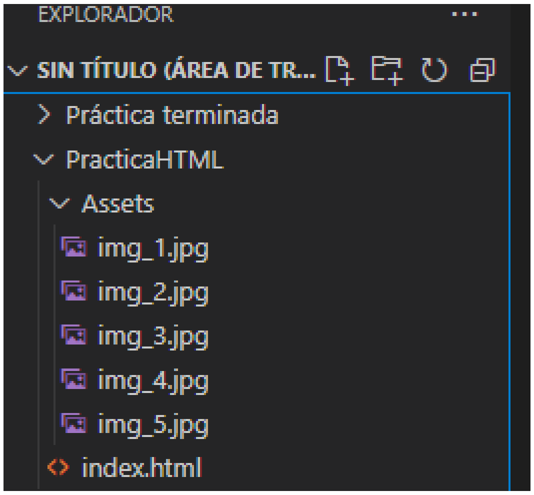
```html
        <section>
            <h3>Imágenes divertidas</h3>
        </section>
```

*Nótese que la nueva etiqueta se añade dentro de la otra etiqueta <section>, no en la que ya habíamos añadido elementos.

Ahora añadiremos las imágenes. Primero crearemos una carpeta “Assets” dentro de la carpeta de proyecto, lo podemos hacer desde VS Code o desde el explorador de archivos.



En esa carpeta agregaremos todas las imágenes que vayamos a cargar. Como todavía no vamos a utilizar hojas de estilo, usaremos imágenes de dimensiones semejantes (es recomendable utilizar imágenes ligeras, de entre 70 a 100kb).



Con las imágenes en la carpeta, ya podremos añadirlas en el HTML Para ello usaremos las etiquetas <figure> e <img> de la siguiente manera (véase apartado 2.5):

```
    <figure>
        <img src="./Assets/img_1.jpg" alt="Descripcion de la imagen"
        <figcaption>Una imagen de dos perritos</figcaption>
    </figure>
```

Nótese como hemos referido la ruta de la imagen en el atributo “src” de la etiqueta <img>. En caso de estar alojada en la web, recordemos que se tendría que utilizar la url de la imagen.

La etiqueta <figcaption> añadirá una pequeña descripción debajo de la imagen, se puede quitar la etiqueta de no querer que salga este texto adicional.

Ahora haremos lo mismo con las cuatro imágenes restantes, cada una con su propia etiqueta <figure>.

```html
        <!-- Sección de imágenes -->
        <section>
            <h3>Imágenes divertidas</h3>

            <figure>
                <img src="./Assets/img_1.jpg" alt="Descripcion de la imagen" />
                <figcaption>Una imagen de dos perritos</figcaption>
            </figure>

            <figure>
                <img src="./Assets/img_2.jpg" alt="Descripcion de la imagen" />
                <figcaption>Una imagen de un perrito blanco</figcaption>
            </figure>

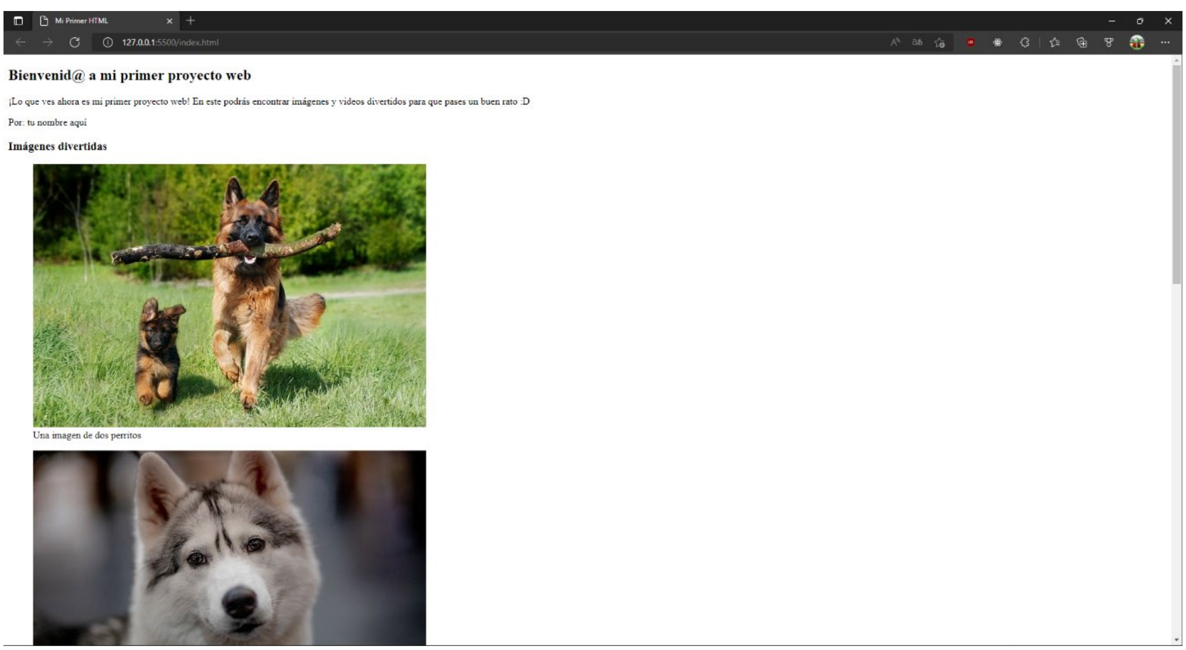
            <figure>
                <img src="./Assets/img_3.jpg" alt="Descripcion de la imagen" />
                <figcaption>Una imagen de un pájaro</figcaption>
            </figure>

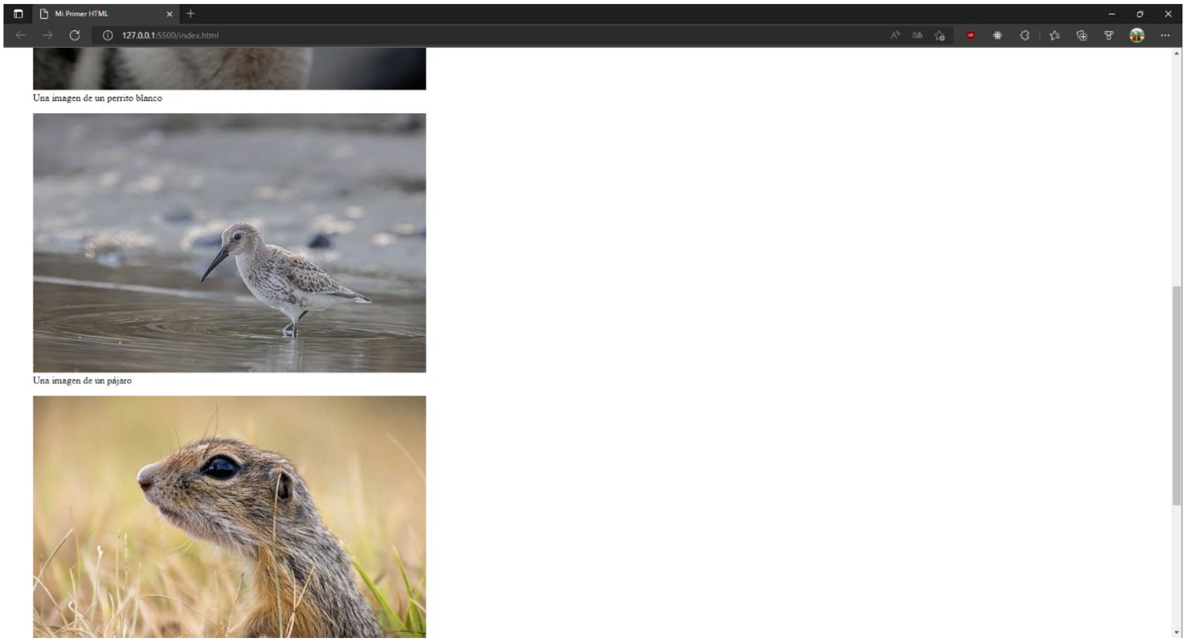
            <figure>
                <img src="./Assets/img_4.jpg" alt="Descripcion de la imagen" />
                <figcaption>Una imagen de una ardilla</figcaption>
            </figure>

            <figure>
                <img src="./Assets/img_5.jpg" alt="Descripcion de la imagen">
                <figcaption>Una imagen de un robot de cartón</figcaption>
            </figure>

        </section>
```

Al guardar nuestro documento, deberíamos de visualizar lo siguiente en el navegador:





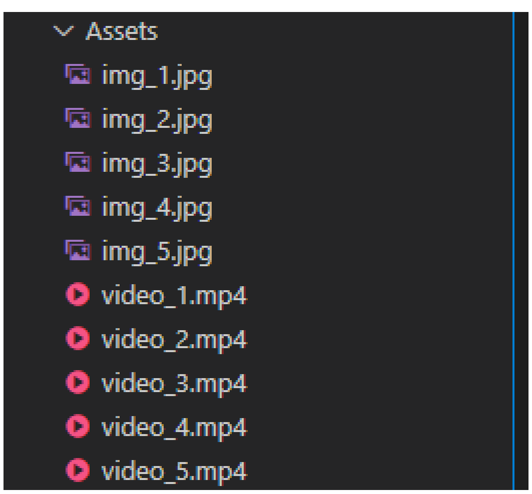
2.6.7 Sección de videos

Empezaremos añadiendo un título a la sección.

Igual que en la sección anterior, usaremos la etiqueta <h3>.

```
        <section>
            <h3>Videos divertidos</h3>
        </section>
```

Ahora añadiremos los videos. Al igual que las imágenes, procuraremos utilizar videos cortos y de dimensiones pequeñas. Podemos agregarlos en la misma carpeta “Assets” que ya habíamos creado anteriormente.



Con los videos en la carpeta, podremos utilizar las etiquetas <video> y <source> para referirlos en el HTML.

```
<figure>
    <video controls preload="auto">
        <source src="./Assets/video_1.mp4" />
    </video>
</figure>
```

*Importante incluir los atributos “controls” y “preload” en la etiqueta <video>.

Haremos lo mismo con los cuatro videos restantes.

```
<!-- Sección de videos -->
<section>
    <h3>Videos divertidos</h3>

    <figure>
        <video controls preload="auto">
            <source src="./Assets/video_1.mp4" />
        </video>
    </figure>

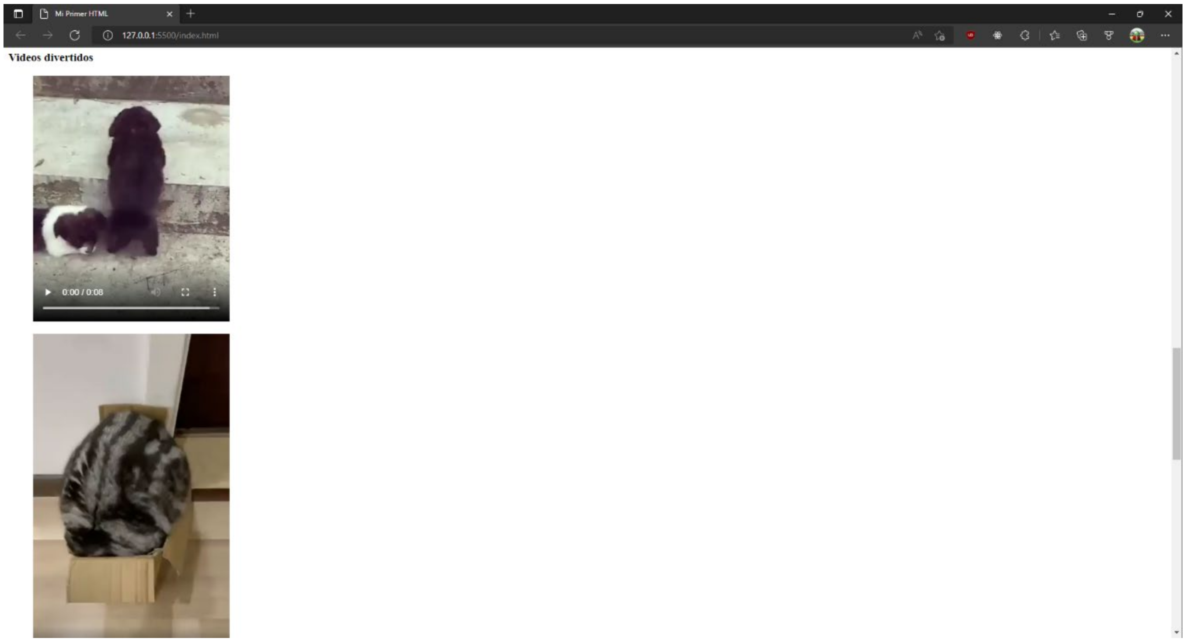
    <figure>
        <video controls preload="auto">
            <source src="./Assets/video_2.mp4" />
        </video>
    </figure>

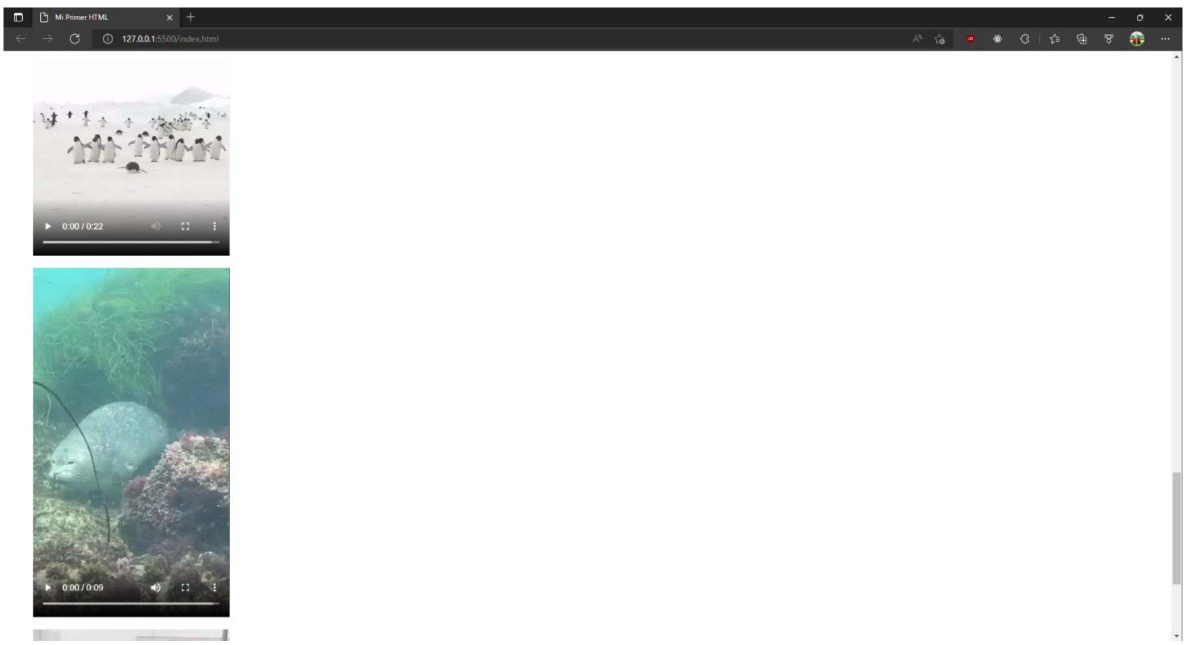
    <figure>
        <video controls preload="auto">
            <source src="./Assets/video_3.mp4" />
        </video>
    </figure>

    <figure>
        <video controls preload="auto">
            <source src="./Assets/video_4.mp4" />
        </video>
    </figure>

    <figure>
        <video controls preload="auto">
            <source src="./Assets/video_5.mp4" />
        </video>
    </figure>
</section>
```

Al guardar el documento, deberíamos de visualizar lo siguiente en el navegador:





¡Hemos terminado! Pero tan solo el documento de referencia.

Recordemos que un HTML es tan solo el “esqueleto” de todo nuestro proyecto web.

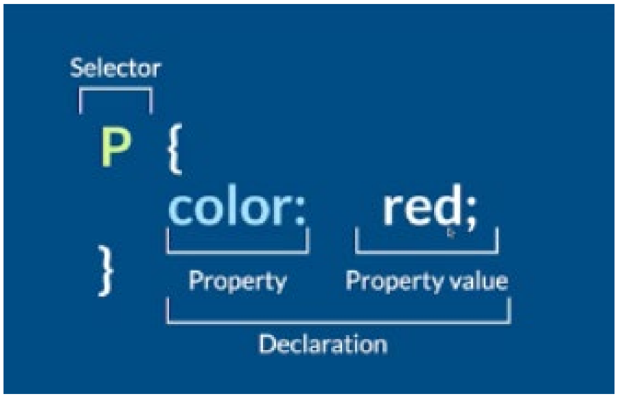
En el siguiente apartado del manual, miraremos como incorporar hojas de estilo para hacer nuestro proyecto mucho más vistoso.

3. CSS

Las hojas de estilo, o archivos .css, son los que aplican los estilos a un sitio web en forma de cascada, leyendo el código de arriba hacia abajo.

3.1 Anatomía de una regla CSS

Una hoja de estilo está compuesta de reglas, estas reglas tienen la siguiente estructura:



El **selector** indicará a que componente del HTML se le aplicarán las propiedades escritas.

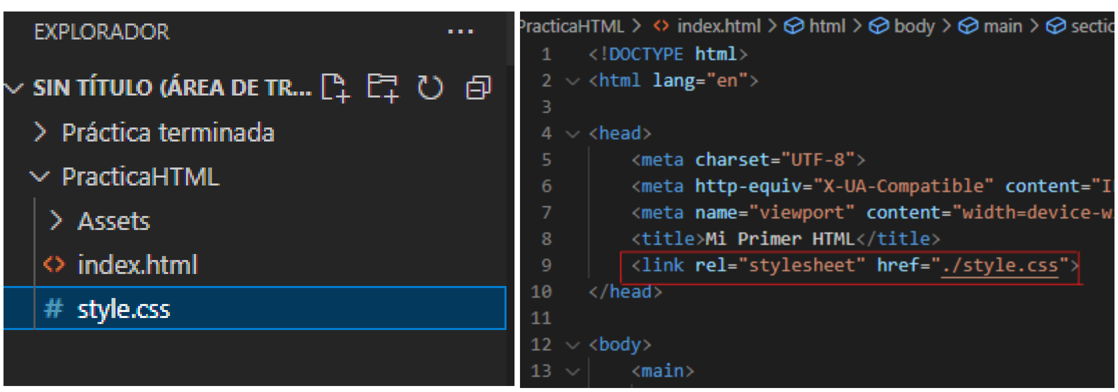
Dentro de la **declaración** tendremos que indicar una **propiedad** y un **valor** a modificar. Las propiedades que podemos alterar son varias, algunas dependerán de que otras propiedades tengan ciertos valores específicos.

3.2 Aplicar un archivo CSS en HTML

Aplicar estilos a los componentes de un HTML puede realizarse de varias maneras.

1. Añadir mediante enlace externo:

Es la forma más recomendada y conveniente de hacerlo, pues los estilos son escritos en un documento aparte. Para hacerlo, se tiene que crear un archivo *.css* junto a nuestro *index.html*, el nombre de este archivo suele ser: *style.css.* Luego lo referimos en el <head>, indicando la ruta correspondiente.



2. Se puede agregar como estilo embebido a una etiqueta HTML:

Si bien la propiedad se aplica correctamente, escribir de esta manera no es para nada practico, sobre todo si pensamos aplicar muchos estilos.

```
<section>
    <h3 style="color: red;">Imágenes divertidas</h3>
```

3. Agregar los estilos dentro de la etiqueta head:

Aplicar estilos de esta manera es prácticamente lo mismo que hacerlo desde un archivo CSS. La diferencia es que ocupamos espacio de nuestro <head> en el HTML, lo que hace el código difícil de leer.

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Primer HTML</title>
    <link rel="stylesheet" href="./style.css">
    <style>
        h3 {
            color: red;
        }
    </style>
</head>
```

3.3 Clases y IDs

Para agregar estilos a una etiqueta de HTML **usamos clases y IDs**. Siendo las clases las más útiles para elementos generales y los IDs para elementos específicos.

3.3.1 Seleccionar etiquetas HTML mediante clases

Para modificar las etiquetas HTML mediante clases deberemos escribir la regla CSS de la siguiente forma:

```
index.html    # style.css
PracticaHTML > # style.css > .nombre_clase
.nombre_clase {
    color: red;
}
```

Donde “nombre_clase” es el nombre que más creamos conveniente para identificar la clase. Nótese que antes del nombre se incluye un “.”, este carácter es el que indica que dicho selector será una clase.

Al guardar la hoja de estilos no veremos cambios en nuestro HTML, así lo tengamos referido en el <head>. Esto es porque toca llamar a la clase en las etiquetas que queramos cambiar, de esta manera:

```
<main>
    <!-- Sección de bienvenida -->
    <section>
        <h1 class="nombre_clase">Bienvenid@ a mi primer proyecto web</h1>
```

Lo anterior cambiará el color de únicamente la etiqueta que tiene la clase, de querer que el color se cambie en otras etiquetas, deberemos igualmente llamar a la clase en cada una de esas etiquetas.

```
<h1 class="nombre_clase">Bienvenid@ a mi primer proyecto web</h1>
<h1>Bienvenid@ a mi primer proyecto web</h1>
<h1 class="nombre_clase">Bienvenid@ a mi primer proyecto web</h1>
```

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Tip adicional: de no querer reutilizar las clases una y otra vez en cada etiqueta, podemos llamar a la clase dentro una etiqueta contenedora, esto aplicará la propiedad en todas las etiquetas "hijas".

```
        <section>
            <div class="nombre_clase">
                <h1>Bienvenid@ a mi primer proyecto web</h1>
                <h1>Bienvenid@ a mi primer proyecto web</h1>
                <h1>Bienvenid@ a mi primer proyecto web</h1>
            </div>
```

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Tip adicional 2: de tener una etiqueta con clase dentro una etiqueta contenedora con clase, podemos referirnos a ella específicamente dentro de CSS.

```
<section>
    <div class="nombre_clase">
        <h1 class="titulo_1">Bienvenid@ a mi primer proyecto web</h1>
        <h1>Bienvenid@ a mi primer proyecto web</h1>
        <h1>Bienvenid@ a mi primer proyecto web</h1>
    </div>
```

```css
.nombre_clase .titulo_1 {
    background-color: blue;
}
```

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

También podemos referirnos a etiquetas dentro una etiqueta con clase, es decir, a todas las etiquetas <h1> dentro del <div class=”nombre_clase”>.

```html
<section>
    <div class="nombre_clase">
        <h1 class="titulo_1">Bienvenid@ a mi primer proyecto web</h1>
        <h1>Bienvenid@ a mi primer proyecto web</h1>
        <h1>Bienvenid@ a mi primer proyecto web</h1>
    </div>
    <h1>Bienvenid@ a mi primer proyecto web</h1>
```

```css

.nombre_clase h1 {
    background-color: blue;
}
```

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Tip adicional 3: una etiqueta puede tener varias clases. Esto es útil en caso de que queramos aplicar una propiedad especifica a una etiqueta con clase.

```html
<section>
    <div class="nombre_clase">
        <h1 class="titulo_1 font_sizeXL">Bienvenid@ a mi primer proyecto web</h1>
        <h1 class="titulo_1">Bienvenid@ a mi primer proyecto web</h1>
        <h1>Bienvenid@ a mi primer proyecto web</h1>
    </div>
```

```css
.nombre_clase {
    color: red;
}

.titulo_1 {
    background-color: blue;
}

.font_sizeXL{
    font-size: 30px;
}
```

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

3.3.2 Seleccionar etiquetas mediante IDs

Para modificar las etiquetas HTML mediante IDs deberemos escribir la regla CSS de la siguiente forma:

```css
#nombre_id{
    color: red;
}
```

Donde “nombre_id” es el nombre que más creamos conveniente para identificar el ID. Nótese que antes del nombre se incluye un “#”, este carácter es el que indica que dicho selector será un ID.

A diferencia de las clases, una etiqueta solo puede tener un ID, y un ID solo puede ser utilizado una vez.

```html
<section>
    <div>
        <h1 id="nombre_id">Bienvenid@ a mi primer proyecto web</h1>
        <h1>Bienvenid@ a mi primer proyecto web</h1>
        <h1>Bienvenid@ a mi primer proyecto web</h1>
    </div>
```

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Bienvenid@ a mi primer proyecto web

Es por esta razón que el ID es recomendable usarlo para referir elementos en JS, no en CSS, si bien podemos usarlos en casos específicos.

3.4 Pseudo-clases y pseudo-elementos

Las pseudo-clases y los pseudo-elementos son selectores que nos permitirán modificar aspectos puntuales de un elemento.

3.4.1 Pseudo-clases

Las pseudo-clases especifican un estado especial a un elemento seleccionado.

Una pseudo-clase bastante interesante es “:hover”, que indica el estado de un elemento cuando el cursor del ratón está sobre este.

```css
.titulo_bienvenida{
    color: blue;
}

.titulo_bienvenida:hover {
    color: red;
}
```

*Nótese que para usar la pseudo-clase se tuvo que referir a la clase del elemento primero, seguido del carácter ":".

Cursor fuera del elemento:

Bienvenid@ a mi primer proyecto web

Cursos encima del elemento:

Bienvenid@ a mi primer proyecto web

En el siguiente enlace se puede encontrar una lista de todas las pseudo-clases: https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-classes

3.4.2 Pseudo-elementos

A diferencia de las pseudo-clases los pseudo-elementos no describen un estado especial, sino que permiten añadir estilos a una parte concreta (a veces muy específica) de un elemento.

Por ejemplo, el pseudo-elemento "::first-letter" se refiere solo a la primera letra de un elemento.

```
.titulo_bienvenida{
    color: blue;
}

.titulo_bienvenida::first-letter {
    color: aqua;
}
```

*Nótese que para usar el pseudo-elemento se tuvo que referir a la clase del elemento primero, seguido de los caracteres "::".

Bienvenid@ a mi primer proyecto web

En el siguiente enlace se puede encontrar una lista de todos los pseudo-elementos: https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-elements

3.5 Medidas en CSS

Para establecer el tamaño de un elemento en CSS, deberemos de considerar las medidas que existen. Las hay de dos tipos:

- Medidas absolutas: es un tipo de medida que no cambia en ningún momento. Los pixeles son de este tipo.

- Medidas relativas: son medidas que son dependientes de otros elementos y que, por ende, si cambian.

3.5.1 Las medidas relativas (EM y REM)

EM

Las medidas Em tomaran el tamaño de fuente de su padre directo. Por ejemplo:

Supongamos que tenemos un párrafo dentro de un <div> con un tamaño de fuente de 16px.

```html
<body>
    <div style="font-size: 16px;">
        <p>Soy un texto :D</p>
    </div>
</body>
```

Soy un texto :D

Ahora supongamos que en nuestro CSS nos referimos a la etiqueta <p> y cambiamos el tamaño de fuente con medida Em.

```css
p {
    font-size: 2em;
}
```

Soy un texto :D

El tamaño de nuestro texto cambiará en consecuencia.

Por tanto, podemos decir que:

1em = 16px

2em = 32px

1.5em = 24px

REM

A diferencia de las medidas Em, las Rem harán referencia a la etiqueta root (base) de nuestro HTML, la cual siempre será <html>. Por tanto, 1rem siempre será igual a 16px (ya que 16px es el tamaño de fuente predeterminado de todo archivo HTML).

Tip adicional: podemos modificar el tamaño de fuente de la etiqueta <html> para utilizar las medidas Rem de la misma forma que los pixeles, así:

1rem = 10px

1.6rem = 16px

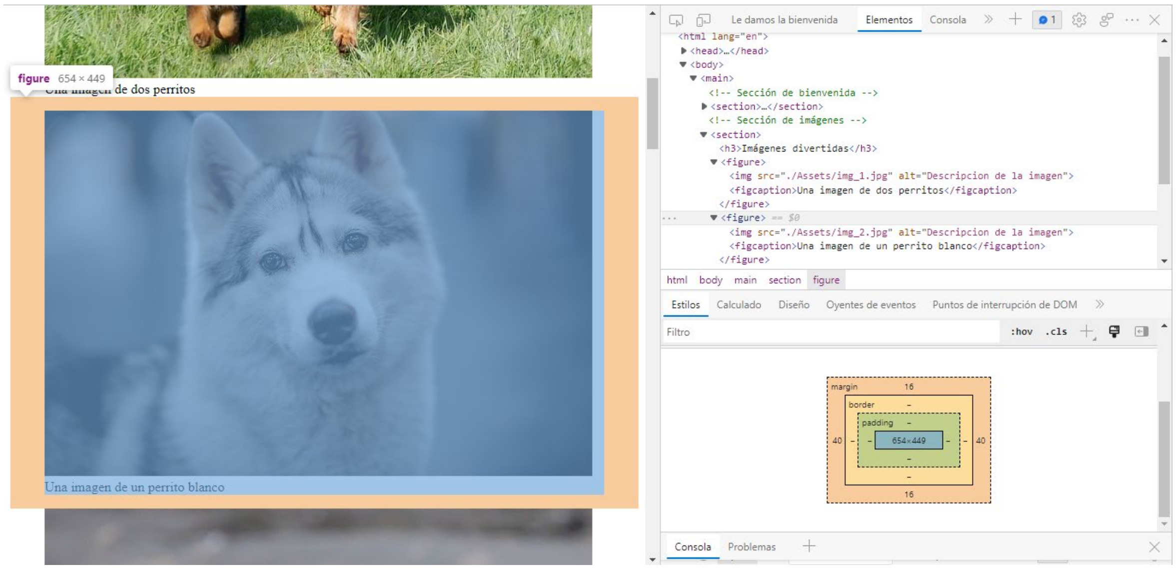
2rem = 20px

Para ello deberemos agregar las siguientes líneas al inicio de nuestro CSS:

```css
html {
    font-size: 62.5%;
}
```

3.6 Modelo de caja

Cada vez que trabajamos con componentes en HTML, estamos manipulando “cajas”, que son contenedoras del contenido que vamos asignando. Esto lo podemos ver de mejor manera con el inspeccionar elementos del navegador.



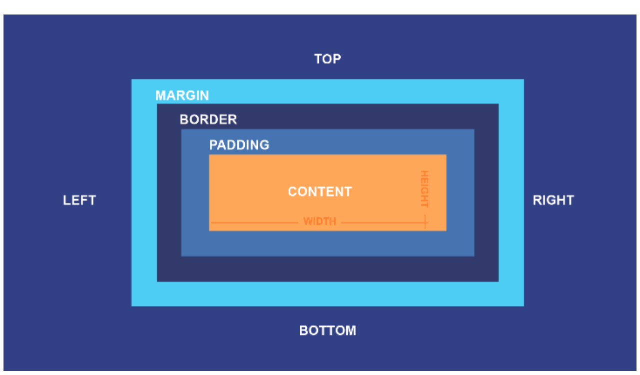
Como podemos observar, son 4 los componentes que conforman cada” caja”, estos son:

- Margin: el espacio externo de la caja.

- Border: el borde del contenido.

- Padding: el espacio interno entre la caja y el contenido.

- Content: el contenido dentro de la caja (texto, imágenes y videos). Son las medidas de este componente las que cambian cuando utilizamos las propiedades width y height.



3.7 Position

Existen distintas propiedades para modificar la posición de los componentes en el HTML, algunas de estas son:

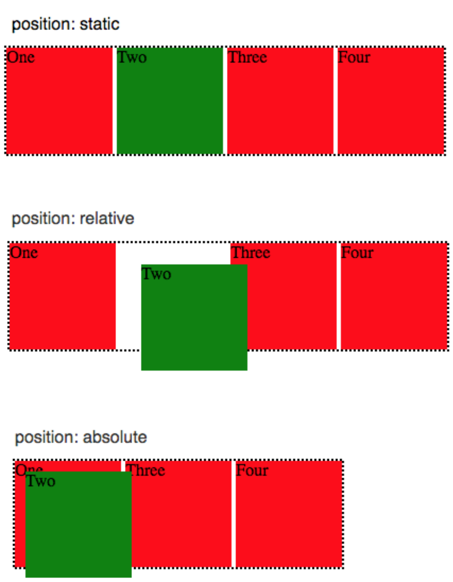
- Static: el componente se queda ubicado en su posición inicial. Es la propiedad que se aplica por defecto.

- Absolute: la posición del componente deja de depender del resto de la página, por tanto, se va a superponer sobre los demás elementos. Los demás elementos ocuparán la posición del componente con posición absoluta.

- Relative: el componente se ubica respecto a su posición original, por tanto, se superpondrá a los demás elementos a pesar de no estar ubicado en esa posición. Los demás elementos conservarán su posición original.

- Fixed: el componente conservará su posición en todo momento a pesar de que se haga scroll.

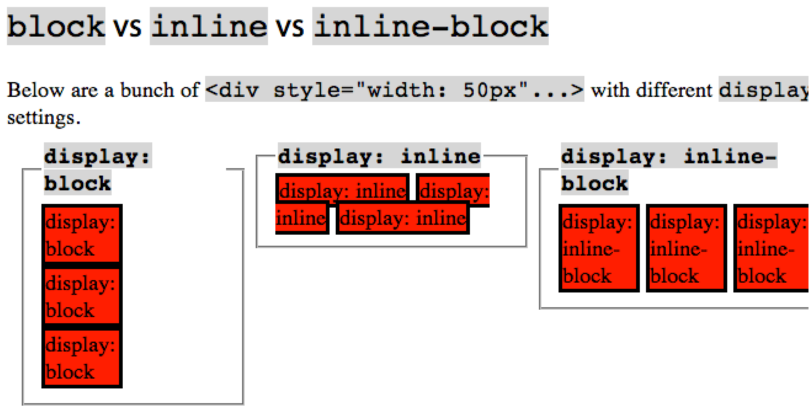
- Sticky: el componente conservará su posición en todo momento cuando el usuario se lo encuentre al hacer scroll.



Tip adicional: el siguiente recurso incluye un ejemplo interactivo que explica las diferencias entre “relative” y “absolute”: https://developer.mozilla.org/en-US/docs/Web/CSS/position. Para ajustar la posición del elemento se tendrán que usar las propiedades “left”, “right”, “top” y “bottom”.

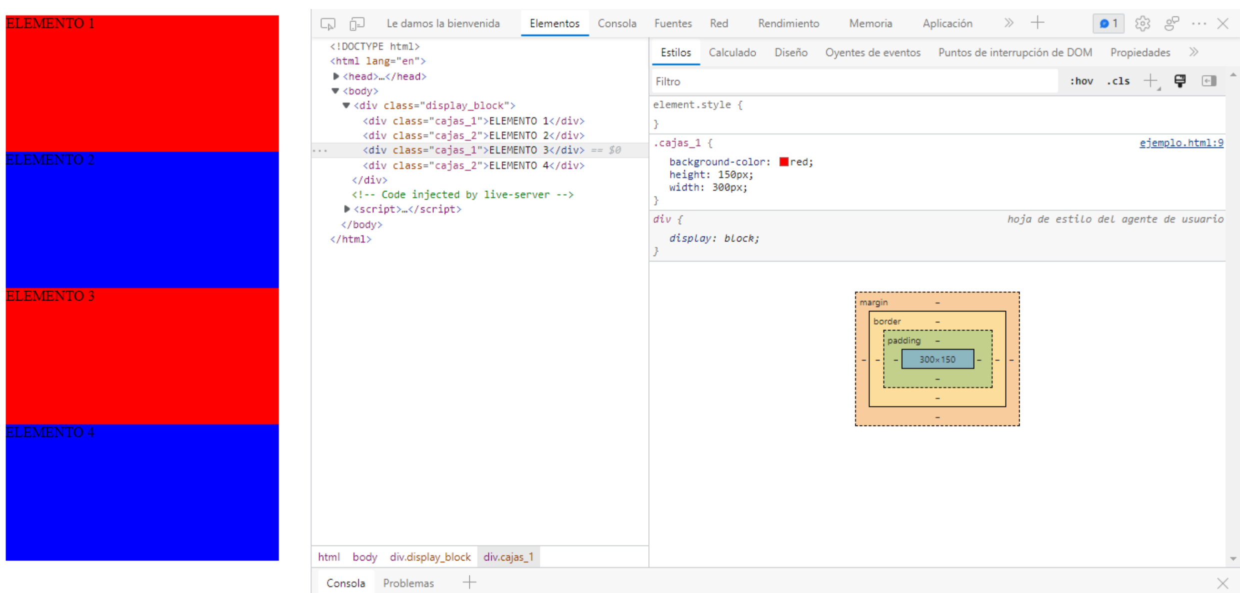
3.8 Tipos de Display

La propiedad Display es la más importante a la hora de manejar el diseño de nuestro proyecto web, pues especificará como un elemento debe mostrarse en pantalla. Entre los tipos de Display más utilizados podemos encontrar los siguientes:



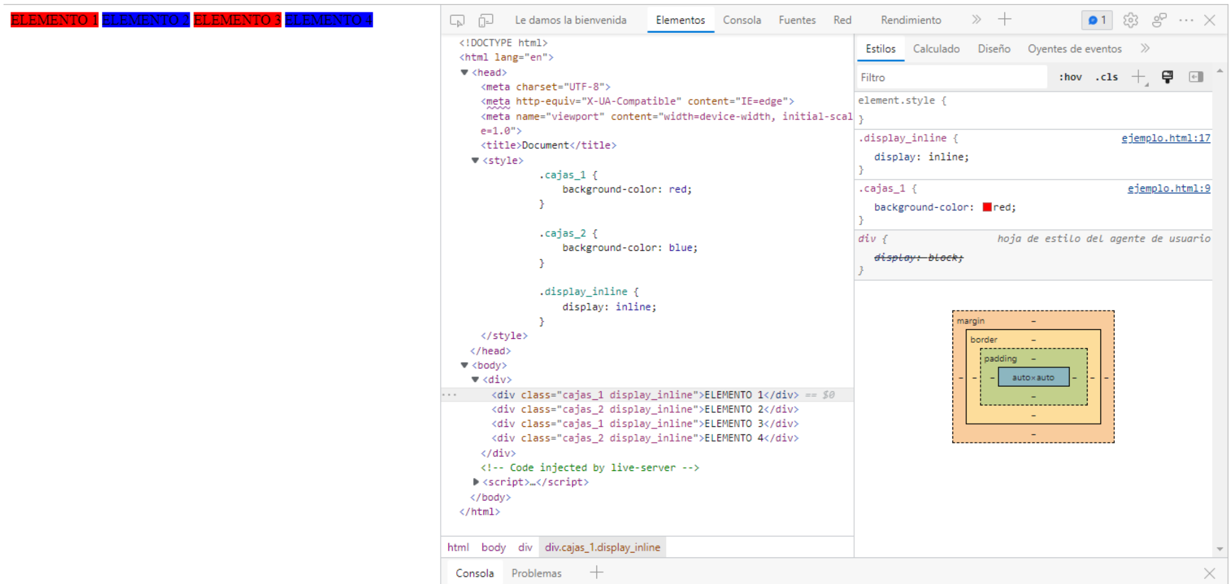
3.8.1 Display block

El Display block usará la totalidad del ancho que tenga sin importar si el contenido tiene o no ese espacio, por tanto, de tener varios elementos, se van a apilar uno encima del otro.



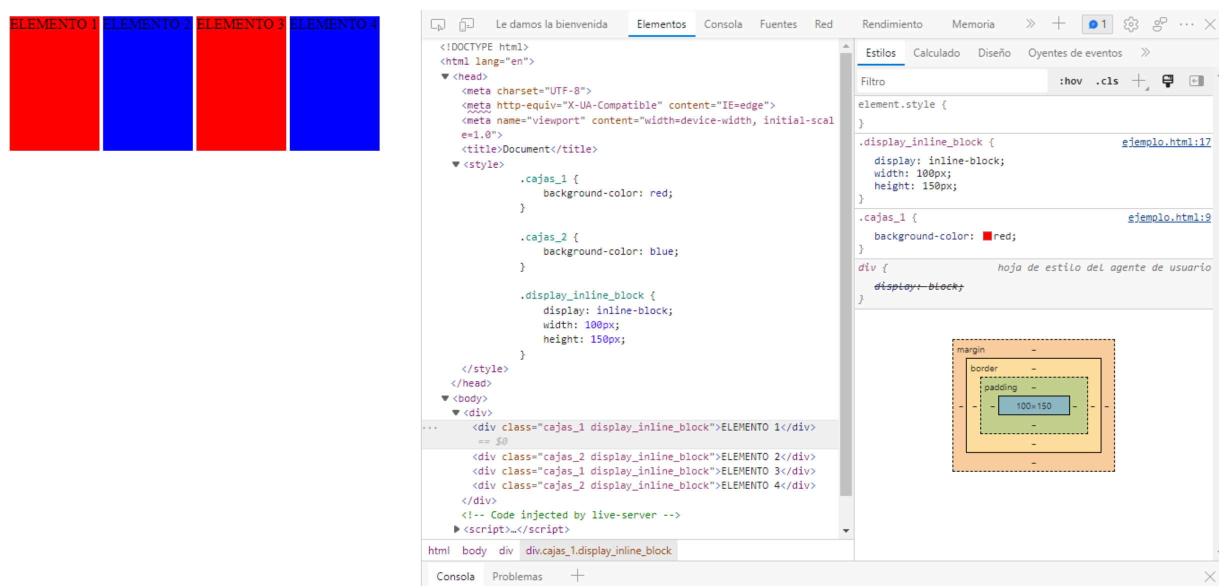
3.8.2 Display inline

El Display inline usará solo el espacio del contenido, por tanto, los elementos se agruparán uno al lado del otro.



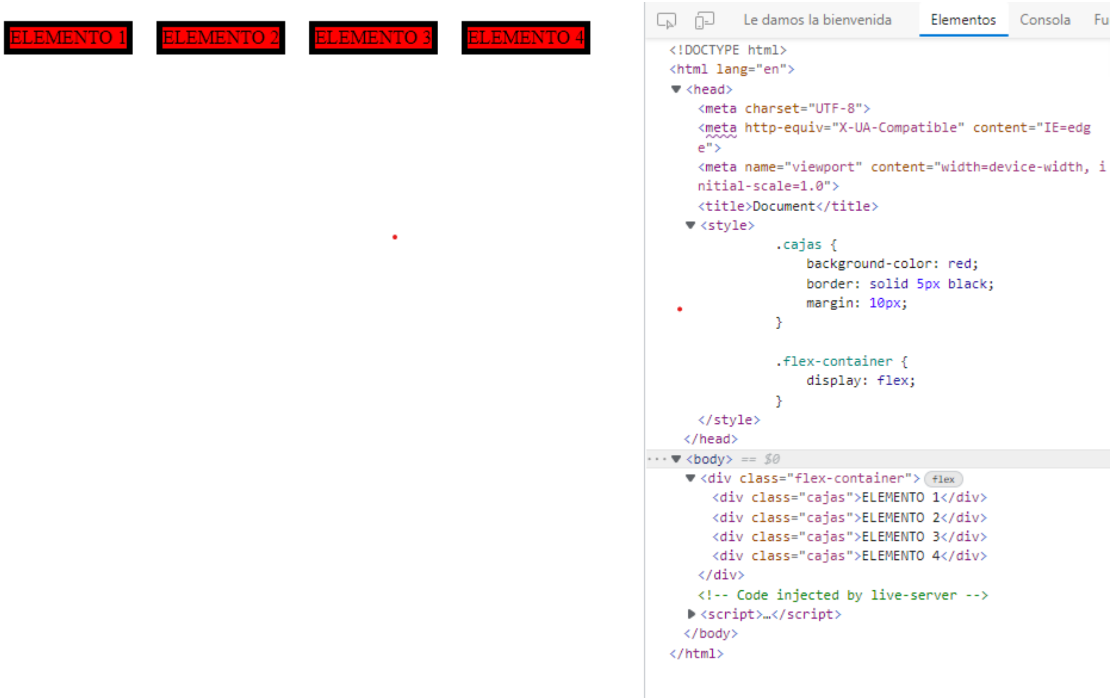
3.8.3 Display inline-block

El display inline-block usará el espacio del elemento, no del contenido.



3.8.4 Display flex

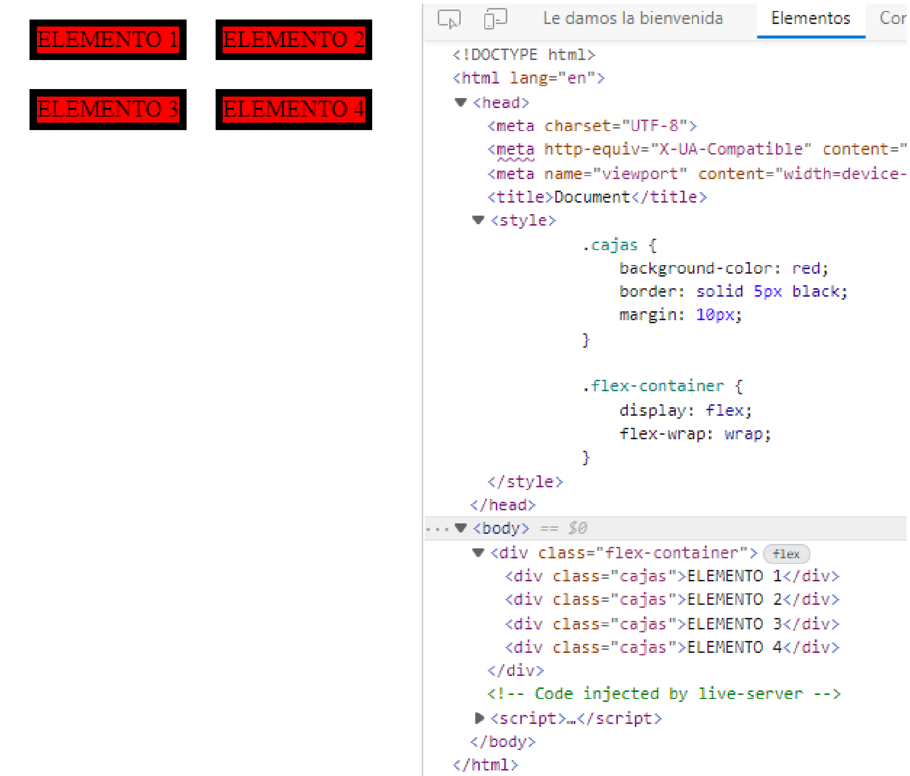
Flex nos permite distribuir el espacio entre los elementos de una manera mucho más fácil, pues distribuye los contenedores a modo de fila o columna. Para usar flex siempre debemos tener un contenedor padre el cual usará el display flex, puede ser un <div>, <section>, <main>, etc.

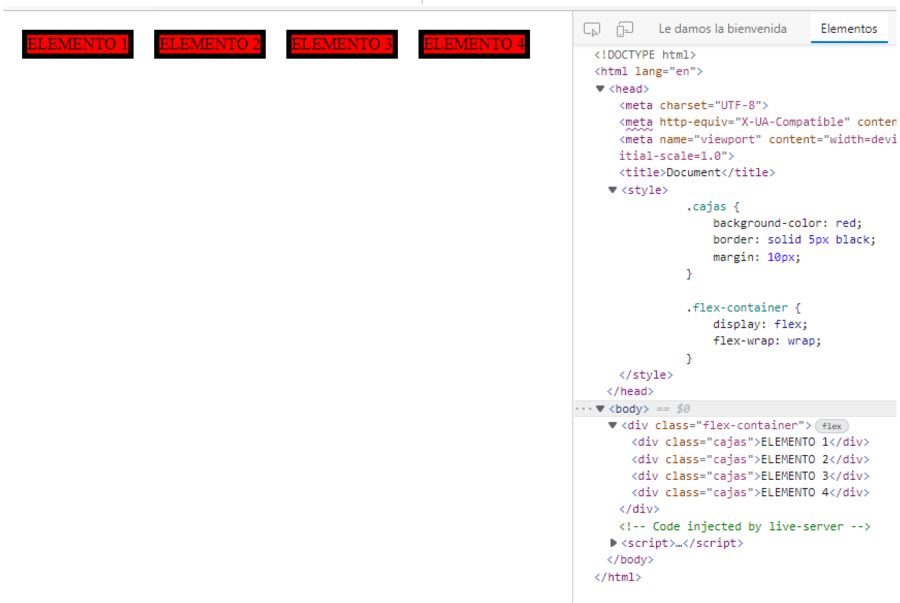


En el ejemplo anterior, flex está utilizando sus ajustes por defecto, por lo que distribuirá los elementos en fila.

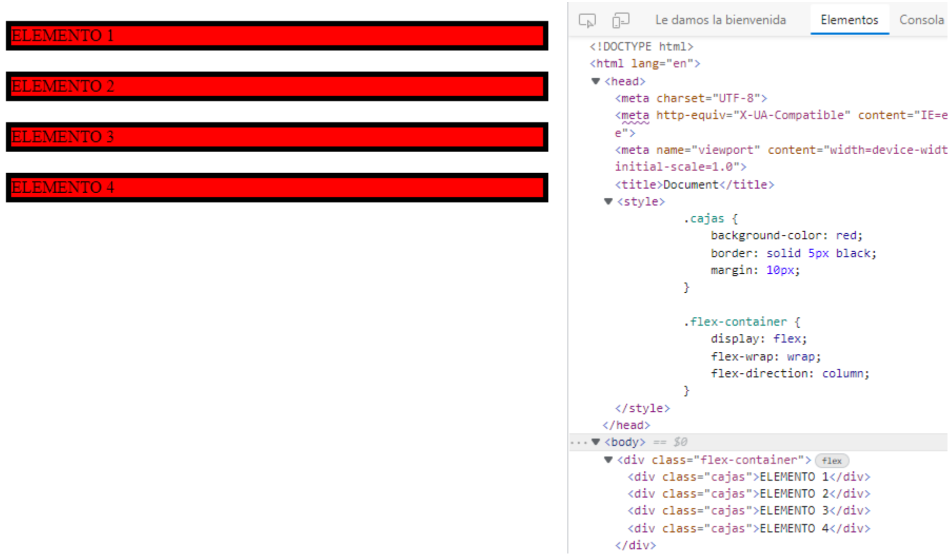
Para ajustar la forma en la que flex distribuye los contenedores deberemos considerar las siguientes propiedades: “flex-wrap” y “flex-direction”.

“flex-wrap” hará que los elementos se ajusten al viewport width del usuario, por tanto, de haber muchos elementos en pantalla, estos irán bajando para formar columnas.



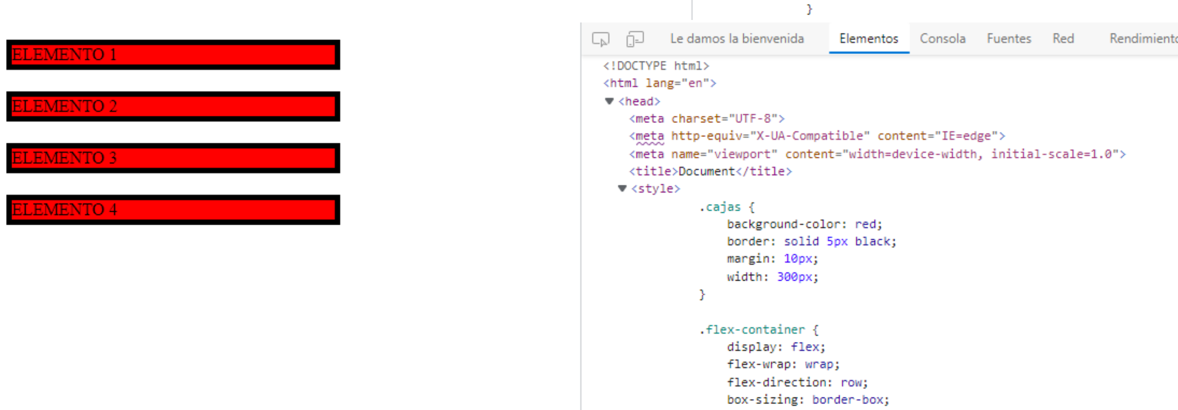


“flex-direction” especificará la dirección o forma en la que los elementos se distribuirán. De querer que los elementos se distribuyan en columnas, se deberá poner el valor “column”.



Tip adicional: otra propiedad de gran utilidad es “box-sizing”. De poner el valor “border-box”, se sumará el padding y el margin con el width del elemento, por tanto, no perderemos contenido ni generaremos un scroll horizontal involuntario.





3.9 Fuentes de letra

Al trabajar con fuentes de letra, tenemos que considerar las fuentes genéricas. Estas son las que vienen preinstaladas en nuestro sistema operativo y las podemos aplicar en un elemento sin necesidad de cargar un archivo de fuente.

De querer utilizar fuentes no genéricas tenemos 2 opciones diferentes:

1. Importar desde archivo CSS:

Para ello deberemos utilizar “@import” seguido de la url de la fuente. Es recomendable ponerlo al inicio del CSS.

```
@import url("https://fonts.googleapis.com/css2?family=Amaranth&display=swap");
```

Luego utilizamos la propiedad “font-family” con el valor de la nueva fuente en el elemento que queramos.

```
.fuente_p {
    font-family: Amaranth, sans-serif;
}
```

```
<div>
    <div>SOY UN TEXTO NORMAL D:</div>
    <div class="fuente_p">SOY UN TEXTO PERSONALIZADO :D</div>
</div>
```

SOY UN TEXTO NORMAL D:
SOY UN TEXTO PERSONALIZADO :D

Importar fuentes con “@import” no es una práctica recomendable, pues afecta el rendimiento general de la página.

2. Añadir desde etiqueta link en HTML:

Para ello deberemos añadir la url de la fuente como link en el <head> del HTML.

```
<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Amaranth&display=swap">
```

Para utilizarla en los elementos, se utiliza la propiedad “font-family”, al igual que en el punto anterior.

3.10 Variables

Las variables en CSS nos permiten almacenar valores de propiedades con la finalidad de no repetir código.

Para declararlas deberemos escribir el selector “:root” al inicio de nuestro CSS. Luego especificaremos el nombre de las variables con los caracteres “--” al inicio, seguido del valor que queremos que guarden.

```
:root {
    --primary-color: #627362;
    --main-font: Amaranth, sans-serif;
}
```

Finalmente podremos usarlas como valores en las propiedades que las requieran.

```css
.fuente_p {
    font-family: var(--main-font);
}

.cajas {
    background-color: var(--primary-color);
    border: solid 5px black;
    margin: 10px;
    width: 300px;
}
```

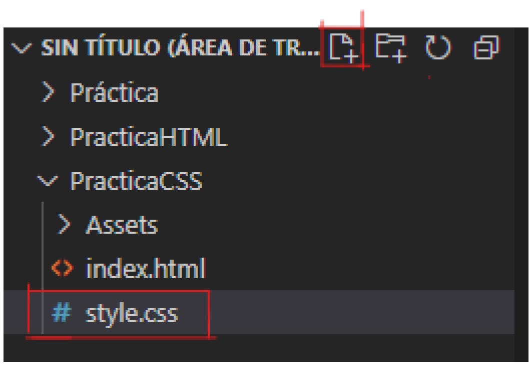
3.11 Poniendo en práctica lo aprendido

Con todo lo visto en la sección anterior, deberías de tener las suficientes herramientas para elaborar tu primera hoja de estilos.

Para practicar, dedicaremos este último apartado para elaborar la hoja de estilos del documento HTML que ya habíamos desarrollado en la sección anterior (véase apartado 2.6).

3.11.1 Creando la hoja de estilos para nuestro proyecto

Vamos a empezar creando la hoja de estilos para nuestro proyecto, es decir, el archivo CSS. Lo podemos hacer desde VS Code.



El nombre puede ser cualquiera, “style” es un nombre genérico.

Luego tendremos que referir el CSS en el HTML, para ello usamos la etiqueta <link> en el <head> (VS Code nos ayuda con algunos atajos para la sintaxis).

```html
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <title>Mi Primer HTML</title>
    link
</head>   link
          link:atom
<body>    link:css                    Emmet Abbreviation
```

```html
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mi Primer HTML</title>
    <link rel="stylesheet" href="style.css">
</head>
```

3.11.2 Organizando nuestros elementos

Antes de empezar a modificar las propiedades de los elementos como color, tamaño... Vamos a organizar el contenido de nuestra página.

Para ello crearemos algunas clases para nuestras etiquetas generales, así podremos ajustar la manera en la que se visualiza el contenido.

```css
.container {
    display: flex;
}

.container section {
    display: flex;
}
```

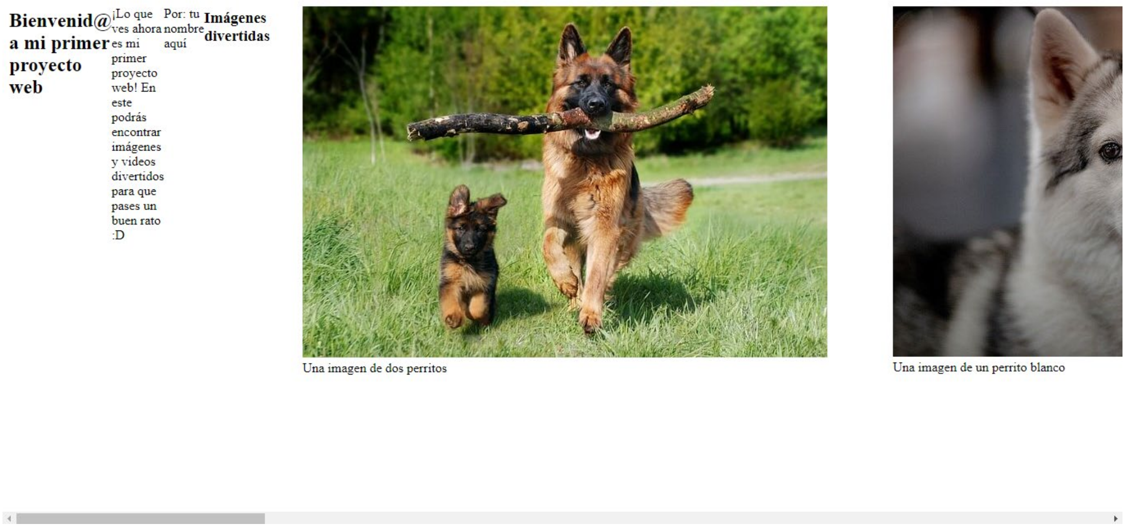
```html
<main class="container">
    <!-- Sección de bienvenida -->
    <section>
        <div>
            <h1>Bienvenid@ a mi primer proyecto web</h
        </div>

        <p>¡Lo que ves ahora es mi primer proyecto web
            que pases un buen rato :D</p>
        <p>Por: tu nombre aquí</p>
    </section>
```

Nótese como hemos creado una clase “container” para la etiqueta <main> y luego, en la siguiente regla, seleccionamos a todas las etiquetas <section> dentro de dicha etiqueta con clase. Con esto nos ahorraremos crear clases adicionales para cada etiqueta <section>. (véase apartado 3.3.1).

Nótese también como hemos utilizado la propiedad “display” con valor “flex”. La gran ventaja de flex es que hará que nuestros elementos se ajusten automáticamente al viewport de cada usuario, por tanto, nuestro contenido siempre se verá completo. (véase apartado 3.8.4).

De guardar nuestro CSS notaremos que nuestra página se ha “girado”.



Esto se debe a que flex por defecto ajusta los elementos en filas. (véase apartado 3.8.4). En nuestro caso particular no queremos eso.

Para solucionarlo usaremos la propiedad “flex-direction”, y le daremos el valor “column”.

```css
.container {
    display: flex;
    flex-direction: column;
}

.container section {
    display: flex;
    flex-direction: column;
}
```

Y listo, devuelta a la normalidad.



Ahora haremos que nuestro contenido se ajuste al centro de la página, para ello, utilizaremos la propiedad “align-items” dentro de la regla “.container section”, y le daremos el valor: “center”.

```css
.container section {
    display: flex;
    flex-direction: column;
    align-items: center;
}
```

¡Y listo!

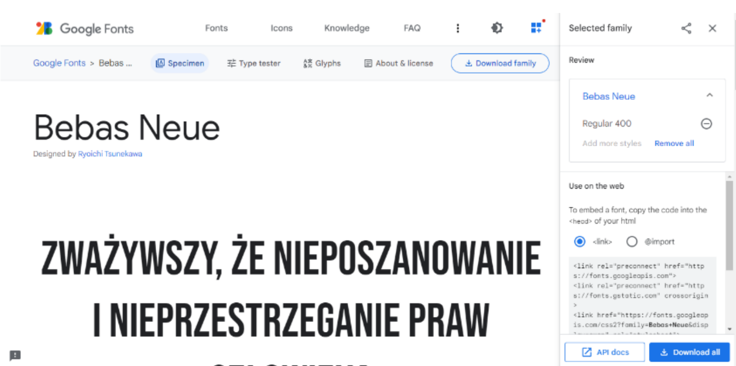


3.11.3 Ajustando el texto

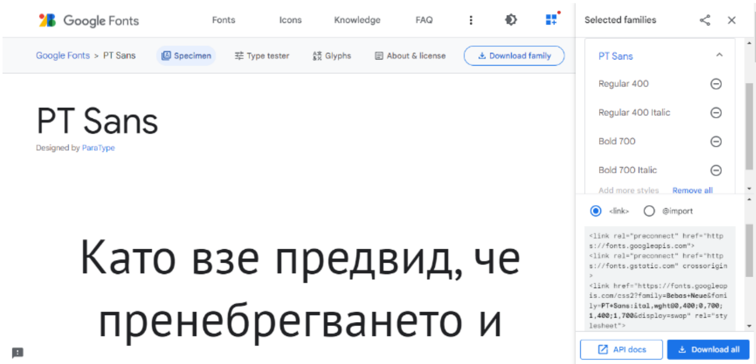
Ahora pasaremos a cambiar el aspecto del contenido en nuestra página. Empecemos por el texto.

En primer lugar, importaremos una fuente de letra desde Google Fonts. Escogeremos una para los títulos y otra para los párrafos.

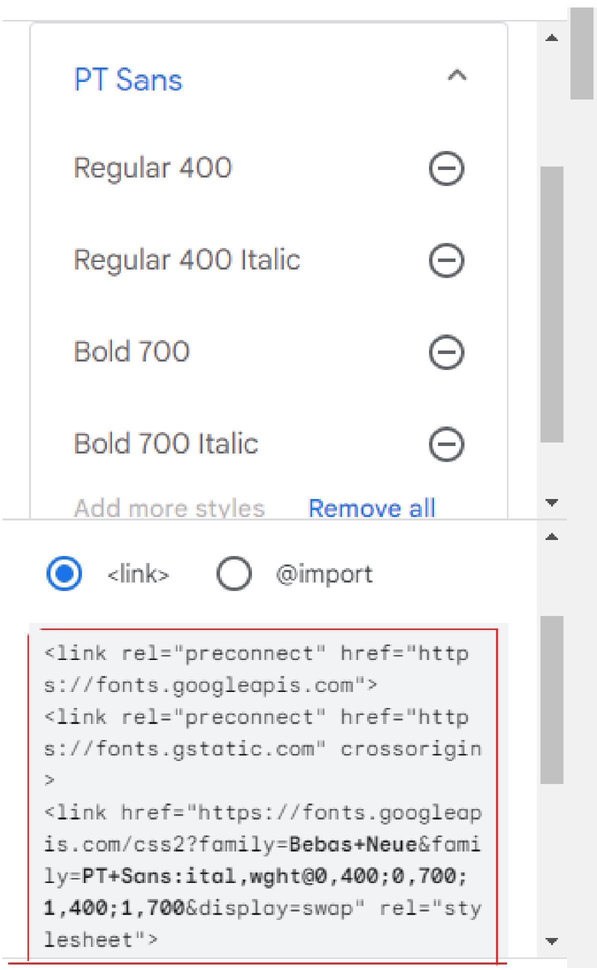
Para los títulos:



Para los párrafos:



Con las fuentes que guardemos, Google Fonts generará automáticamente el código que deberemos añadir al <head>.





Con las fuentes cargadas, vamos a crear dos variables que guarden como valor cada tipo de fuente, así no tendremos que repetir código cada vez que queramos aplicarlas en algún elemento. (véase apartado 3.10).

```
:root{
    --title-font: 'Bebas Neue', cursive;
    --text-font: 'PT Sans', sans-serif;
}
```

*Lo que se pone de valor en las variables son las familias de las fuentes que hemos cargado. Google Fonts genera ese código automáticamente con las fuentes que especifiquemos.

Por temas de accesibilidad, trabajaremos el tamaño de los textos con medidas Rem (véase apartado 3.5.1). Al utilizar medidas relativas, le damos posibilidad al usuario de ajustar el tamaño del texto desde el navegador.

Ajustaremos el tamaño predeterminado del HTML para poder trabajar las medidas Rem de igual forma que los pixeles. (véase apartado 3.5.1).

```
html {
    font-size: 62.5%;
}
```

Ahora ajustaremos las demás propiedades de los textos.

```
.title {
    font-family: var(--title-font);
    font-size: 6rem;
}

.sub-title {
    font-family: var(--title-font);
    font-size: 4rem;
}

p {
    font-family: var(--text-font);
    font-size: 2rem;
    text-align: center;
}
```

Debido a que queremos mantener el mismo tamaño y fuente en los párrafos, hemos utilizado de selector a la etiqueta <p>. Para los títulos si hemos creado dos clases independientes.

La propiedad “text-align” con valor “center” alineará el texto de los párrafos al centro.

3.11.4 Aplicando una paleta de colores

Los colores son un aspecto muy importante en todo sitio web. Siempre es recomendable tener preestablecida una paleta de colores. Para nuestro sitio web hemos escogido los siguientes (usaremos el código HEX).

- Color primario: #DAF7A6

- Color secundario: #FFC300

Para facilitar las cosas, crearemos variables adicionales para guardar estos valores.

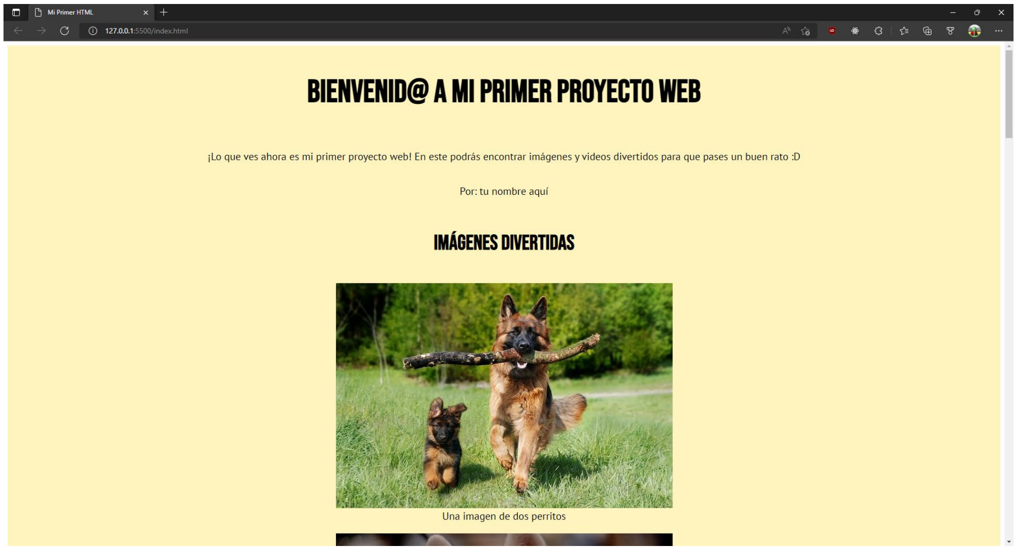
```
:root {
    --title-font: 'Bebas Neue', cursive;
    --text-font: 'PT Sans', sans-serif;
    --primary-color: #DAF7A6;
    --secondary-color: #FFC300;
}
```

Ahora aplicaremos los valores en las propiedades de los elementos que queramos. En nuestro caso, cambiaremos el “background-color” de todo el documento.

```
.container {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: 100%;
    background-color: var(--primary-color);
}
```

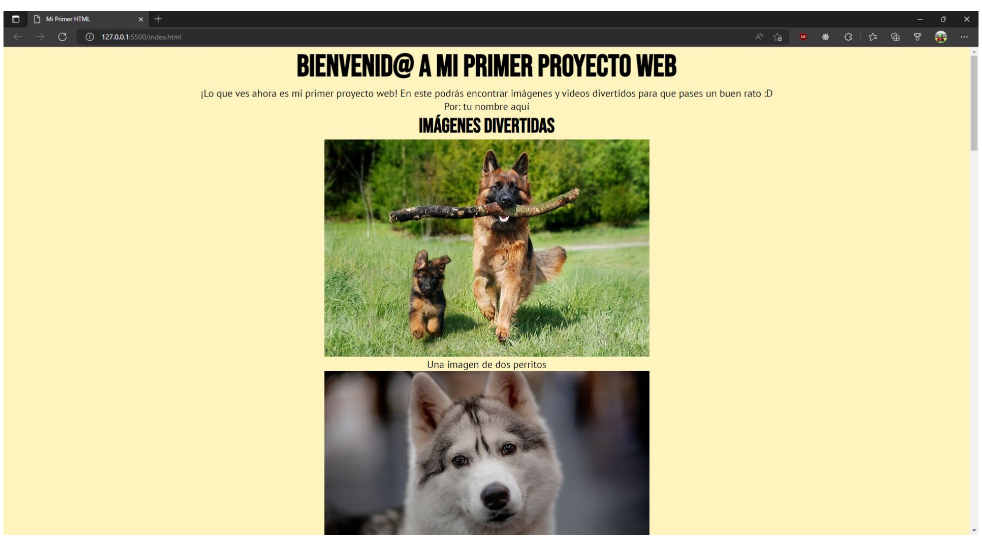
*Hemos aplicado la propiedad en la clase “container” debido a que es la etiqueta “padre” de todo el contenido.

*Nótese como hemos añadido las propiedades “width” y “height” con valores “100%”, esto ajustará todo el contenedor al viewport del usuario, por tanto, todo el color del fondo cambiará.



Al guardar la hoja de estilos, notaremos que hay una pequeña margen a los bordes de la página. Esto es porque HTML aplica un valor predeterminado. Para quitar dicha margen crearemos una regla al inicio del CSS con el selector "*" (que se refiere a todo el documento), y utilizar la propiedad "box-sizing" con el valor: "border-box". También tendremos que definir las propiedades margin y padding, ambas con valor "0".

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```



3.11.5 Estilizando las imágenes

Ahora estilizaremos nuestra galería de imágenes. Para ello, lo primero que haremos será añadir un marco a cada imagen. La propiedad "border" nos puede servir.

Existen distintas opciones a la hora de personalizar el borde. VS Code nos ayudará y mostrará algunas de ellas. En nuestro caso lo pondremos como “solid” y de 10 px de tamaño.

También vamos a añadir una sombra con la propiedad “box-shadow”.

```
img {
    border: solid 10px var(--secondary-color);
    box-shadow: 0px 6px 10px 1px rgba(0,0,0,1);
}
```

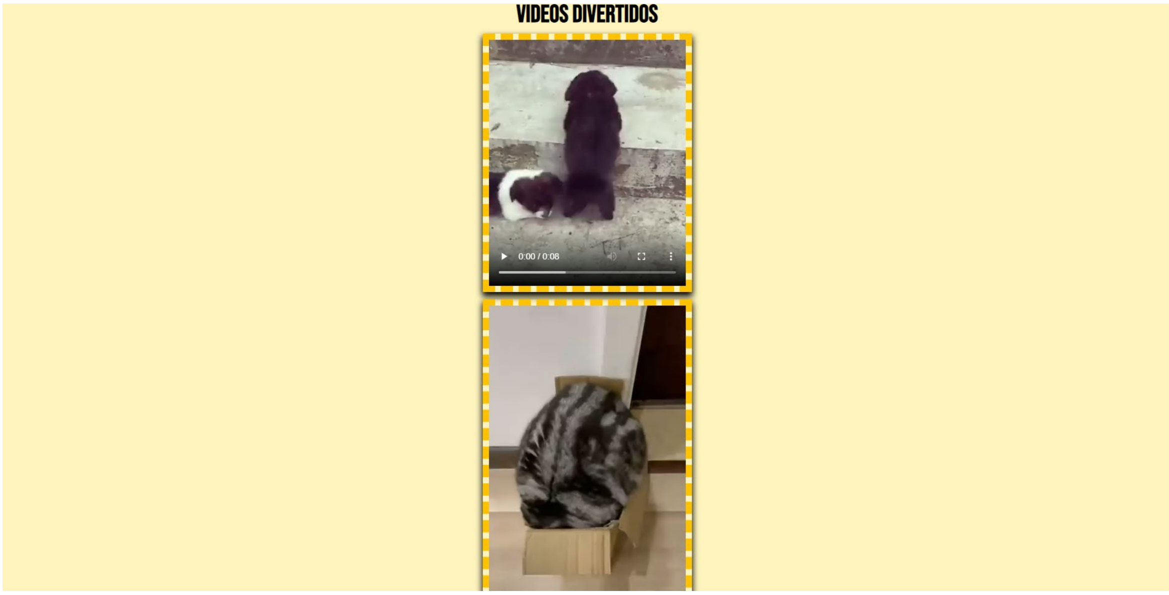
*Existen generadores que nos ayudarán a escribir estas propiedades, una que recomendamos es: https://cssgenerator.org/



3.11.6 Estilizando los videos

Ahora añadiremos un marco a los videos. Usaremos la propiedad “border”, pero esta vez con el valor “dashed”.

```
video {
    border: 10px dashed var(--secondary-color);
    box-shadow: 0px 6px 10px 1px rgba(0,0,0,1);
    margin: 5px;
}
```



También tendremos la opción de distribuir los videos en fila. Para ello crearemos un <div> en el HTML, donde contendremos todas las etiquetas <video>. Luego crearemos una clase para ese <div> que distribuya los elementos con ayuda de flex.

```html
<section>
    <h3 class="sub-title">Videos divertidos</h3>
    <div class="videos-container">
        <figure>
            <video controls preload="auto">
                <source src="./Assets/video_1.mp4" />
            </video>
        </figure>

        <figure>
            <video controls preload="auto">
                <source src="./Assets/video_2.mp4" />
            </video>
        </figure>

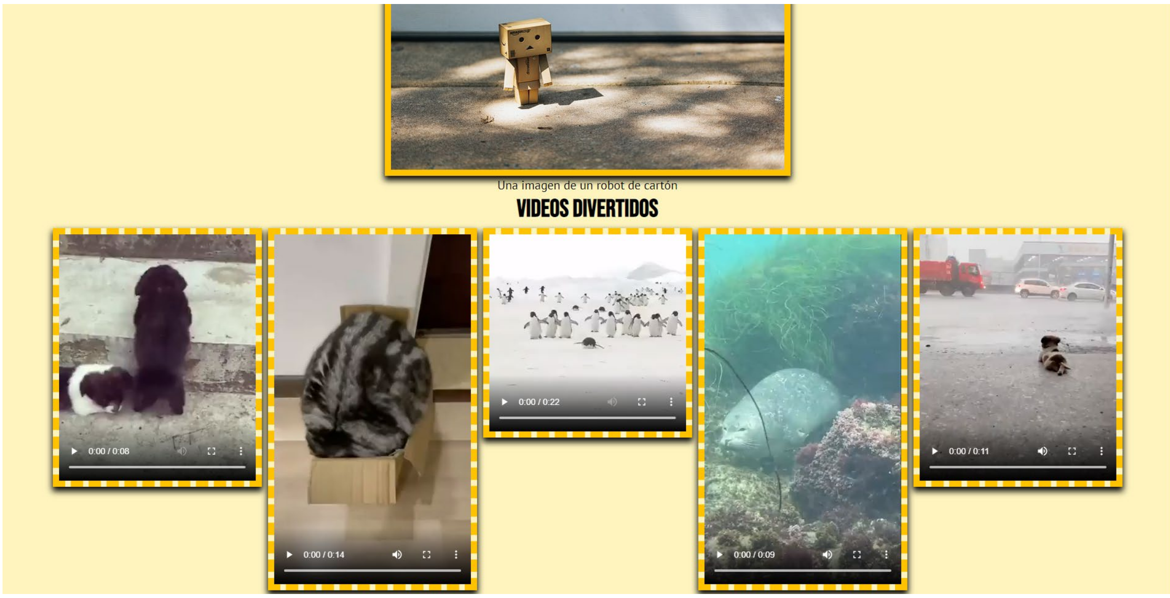
        <figure>
            <video controls preload="auto">
                <source src="./Assets/video_3.mp4" />
            </video>
        </figure>

        <figure>
            <video controls preload="auto">
                <source src="./Assets/video_4.mp4" />
            </video>
        </figure>

        <figure>
            <video controls preload="auto">
                <source src="./Assets/video_5.mp4" />
            </video>
        </figure>

    </div>
</section>
```

```css
.videos-container {
    display: flex;
    flex-direction: row;
}
```



Y así habremos terminado nuestra hoja de estilos.

CSS nos permite estilizar nuestra página de incontables maneras. Experimenta con las propiedades y estiliza tu proyecto como quieras.